Bundesamt
für Sicherheit in der
Informationstechnik

Eine Studie im Auftrag des
Bundesamtes für Sicherheit in der Informationstechnik (BSI)

TSS Studie

Einführung und Analyse des OSS TCG Software Stack TrouSerS und Werkzeuge in dessen Umfeld

Version 1.0

Autoren:

Marcel Selhorst, Christian Stüble, Felix Teerkorn

Sirrix AG security technologies
Lise-Meitner-Allee 4
44801 Bochum
Deutschland

Begleitung dieser Studie:

Florian v. Samson, Daniel Nowack
Bundesamt für Sicherheit in der Informationstechnik (BSI)
Postfach 200363
53133 Bonn
Deutschland

Lizenzbedingungen:

Diese Arbeit wird unter den Lizenzbedingungen der Creative Commons Public Lizenz “Namensnennung - Keine Bearbeitung 3.0 (CCPL-by-ND 3.0)” veröffentlicht. Im Detail bedeutet dies:

- Vervielfältigen. Sie dürfen das Werk vervielfältigen, verbreiten und öffentlich zugänglich machen.
- Namensnennung. Sie müssen den Namen des Autors/Rechteinhabers in der von ihm festgelegten Weise nennen (wodurch aber nicht der Eindruck entstehen darf, Sie oder die Nutzung des Werkes durch Sie würden entlohnt).
- Keine Bearbeitung. Dieses Werk darf nicht bearbeitet oder in anderer Weise verändert werden.

Die komplette Lizenzvereinbarung befindet sich im Anhang D.

Inhaltsverzeichnis

1. Einführung....5
1.1 Begriffe und Definitionen....7
2. High-Level-Design und Interaktion der Komponenten....8
2.1. GRUB und seine TC-Erweiterungen....8
2.1.1. TrustedGRUB....10
2.1.1.1. Installation und Konfiguration....11
2.1.1.2. Zusätzliche Funktionalitäten....12
2.1.1.3. Qualität der Implementierung....13
2.1.1.4. Zukünftige Arbeit....14
2.1.2. GRUB-IMA....14
2.1.2.1. Installation und Konfiguration....15
2.1.2.2. Zusätzliche Funktionalitäten....16
2.1.2.3. Qualität der Implementierung....16
2.1.3. OSLO....17
2.1.3.1. Installation und Konfiguration....17
2.1.3.2. Qualität der Implementierung....17
2.1.4. Bootstrapping....18
2.2. Linux TDD....20
2.2.1. High-Level-Design....20
2.2.2. Qualität der Implementierung....23
2.3. TrouSerS-TSS....23
2.3.1. High-Level-Design....24
2.3.2. TSS und Linux TDD....25
2.3.3. Installation und Konfiguration....27
2.3.4. Qualität der Implementierung....27
2.4. Die TrouSerS TPM-Tools....28
2.4.1. Installation und Konfiguration....28
2.4.2. High-Level-Design....28
2.4.3. Funktionalität....29
2.4.4. Sealing mit den TPM-Tools....31
2.4.5. Qualität der Implementierung....32
2.5. TPM-Manager....34
2.5.1. Installation und Konfiguration....34
2.5.2. High-Level-Design....35
2.5.3. „Take Ownership“ mit dem TPM-Manager....37
2.5.4. Qualität der Implementierung....38
2.5.5. Zukünftige Arbeiten....39
2.6. OpenSSL-TPM-Engine....39
2.6.1. OpenSSL-Engine-Schnittstelle....39
2.6.2.OpenSSL-Engine-Aktivierung....40
2.6.3. Die TPM-Engine....41
2.6.3.1. Konfiguration und Installation....41
2.6.3.2. Qualität der Implementierung....42
2.6.4. TPM-basierte „Transport Layer Security“ mit der OpenSSL-TPM-Engine....43
2.6.5. Zukünftige Entwicklungen....44

3. Einhaltung der Spezifikationen und Interoperabilität....45
3.1. Compliance-Analyse....45
3.1.1. Bootloader Compliance....45
3.1.1.1. TrustedGRUB....45
3.1.1.2. GRUB-IMA....47
3.1.1.3. OSLO....49
3.1.2. TrouSerS TSS....49
3.1.3. OpenSSL-TPM-Engine....50
3.1.4. TPM-Manager....51
3.2. Testumgebung und Testfälle....53
3.2.1. Testumgebung....53
3.2.1.1. Security-Enhanced Linux (SE-Linux)....53
3.2.1.2. Der Xen-Hypervisor....55
3.2.1.3. Der Turaya-Sicherheitskern....55
3.2.2. Testfälle....57
3.3. Interoperabilitätstests....58
3.3.1. SE-Linux-basierte Architektur....58
3.3.2. Xen-basierte Architektur....61
3.3.3. Turaya-basierte Architektur....62
4. Zusammenfassung....64
4.1. „Trusted Computing“-Bausteine....64
4.2. „Trusted Computing“-Architekturen....64
4.3. Offene Fragen....66
4.3.1. Kryptographische API mit TC-Unterstützung....66
4.3.2. Managementlösungen....67
4.3.3. Infrastrukturelle Lösungen....68
Anhang....70
Anhang A. TPM_Unseal....71
Anhang B. Patchset für TPM Tools....72
Anhang C. SE-Linux Installation....74
Anhang D. Creative Commons Lizenz by ND....77
Literaturverzeichnis....83
Abkürzungsverzeichnis....86
Glossar....88

1. Einführung

Die Bedeutung von „sicheren Betriebssystemen“ hat in den letzten Jahren in vielerlei Hinsicht stark zugenommen. Das liegt hauptsächlich daran, dass viele kommerzielle und private Anwendungen für IT-Systeme entwickelt werden, die über einen direkten Zugang zum Internet verfügen. Somit sind diese IT-Systeme und damit auch die auf ihnen laufenden Anwendungen ständig durch Angriffe von Schadprogrammen wie Würmern und trojanischen Pferden bedroht.

Die neuesten Entwicklungen im Bereich „Trusted Computing (TC)“, insbesondere die durch die Trusted Computing Group (TCG) spezifizierten Hard- und Softwareerweiterungen wie Trusted Platform Module (TPM) und TCG Software Stack (TSS), versprechen erhebliche Fortschritte im Bereich der sicheren Betriebssysteme. Zur Absicherung eines IT-Systems ist aber die ausschließliche Verwendung von Funktionen, wie sie durch ein TPM und den TSS bereitgestellt werden, nicht ausreichend. Vielmehr müssen auch Middleware und Anwendungen entsprechend erweitert werden, um die oben genannten Funktionen gewinnbringend nutzen zu können.

Eine Reihe von relevanten Softwarekomponenten im „Trusted Computing“-Umfeld sind bereits seit längerer Zeit als Open Source Software (OSS) erhältlich, allerdings handelt es sich hierbei im Wesentlichen um grundlegende Bausteine. Darauf aufbauende Dienste und Anwendungen, die ebenfalls OSS sind, beginnen sich erst in letzter Zeit zu etablieren.

Um Anhaltspunkte zu geben inwiefern die Entwicklung von TC-basierten Anwendungen möglich ist, bietet diese Studie einen Überblick über derzeit bestehende OSS Software, welche „Trusted Computing“-Technologien unterstützen. Zusätzlich wird untersucht, ob die hier vorgestellten Softwarekomponenten ihren Spezifikationen genügen, ob sie untereinander kompatibel sind und inwiefern Teile der Software Mängel aufweisen. Dazu umfasst die vorliegende Studie folgende Themen:

1. Eine Analyse, inwieweit die wesentlichen OSS Komponenten im Zusammen hang mit „Trusted Computing“ bereits den von der TCG vorgegebenen Spezifikationen entsprechen. Insbesondere soll dabei die Konformität der Schnittstellen als auch die Vollständigkeit der Implementierung untersucht werden.
2. Eine Untersuchung der Architektur der einzelnen Komponenten sowie des möglichen Zusammenspiels der existierenden Komponenten im Rahmen einer TC-Gesamtarchitektur. Dabei wird ebenfalls die verwendete Programmiersprache berücksichtigt, was im Fall von gemeinsam genutzten Bibliotheken eine besondere Bedeutung hat.
3. Das Aufzeigen von möglichen zukünftigen „Trusted Computing“-Architekturen, und wie sich die existierenden Komponenten in dieses Bild einfügen. Dabei wird besonders analysiert, ob und welche Komponenten und Funktionalitäten insbesondere für die Erstellung sicherer Anwendungen noch fehlen.

4. Eine Evaluierung der Interoperabilität zu und Integration in die aktuellen Sicherheitsund Virtualisierungslösungen SE-Linux, Xen und Turaya, einschließlich einer Liste von berücksichtigten Testfällen.

1.1 Begriffe und Definitionen

Die in dieser Studie verwendeten Begriffe und Definitionen sind im Glossar definiert, während Abkürzungen im Abkürzungsverzeichnis am Ende des Dokuments zu finden sind. Weiterhin werden in der vorliegenden Studie die folgenden Begriffe verwendet:

Trusted Computing. In diesem Zusammenhang wird der Begriff „Trusted Computing“ in Anlehnung an die Definition durch die TCG verwendet, d. h. es werden Komponenten und Mechanismen beschrieben, die in der Spezifikation der TCG definiert oder zumindest dazu kompatibel sind

Linux. Falls nicht explizit darauf hingewiesen wird, beschreibt der Begriff Linux“ den Linux Kern. Falls ein gesamtes Betriebssystem, welches den Linux Kern verwendet, identifiziert werden soll, wird der Begriff „GNU/Linux Betriebssystem“ verwendet.

Trusted Boot. Die Sicherheitseigenschaft einer Bootstraparchitektur gemäß des Bootstrap-Modells der TCG (d. h. Alle am Bootvorgang beteiligten Komponenten werden vor der Ausführung derart gemessen, dass entfernte Stellen sie verifizieren können).

Secure Boot. Die Sicherheitseigenschaft einer Bootstraparchitektur, die nur eine vordefinierte Konfiguration an Softwarekomponenten bootet. Wenn die Software geändert wurde, wird der Bootvorgang unterbrochen.

2. High-Level-Design und Interaktion der Komponenten

Dieses Kapitel stellt die bereits existierenden Open-Source-Komponenten vor, die „Trusted Computing“ unterstützen und so den Aufbau einer TC-Plattform ermöglichen. Dies umfasst den TPM-Manager, eine graphische anwenderbasierte TPM-Management Komponente, die TPM-Tools des TrouSerS-Projektes, die OpenSSL TPM-Engine, den TrouSerS TSS, den Linux-TPM Device Driver (TDD), sowie verschiedene TCG-Erweiterungen des GRUB Bootloaders.

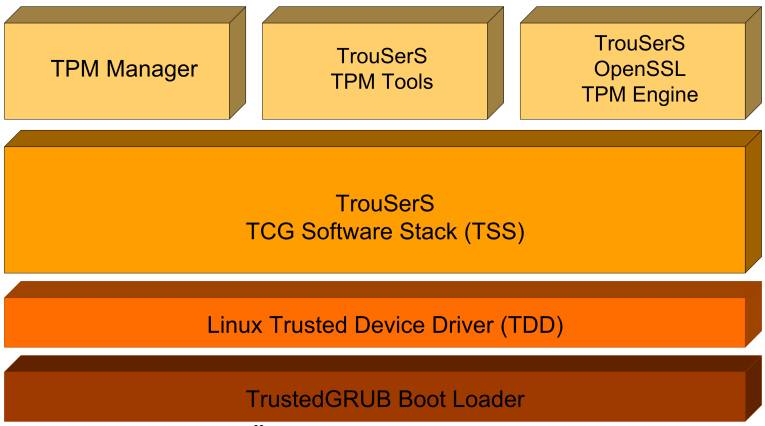


Abbildung 2.1.: Übersicht der in Kapitel 2 erläuterten Softwarekomponenten.

Die zu analysierenden Komponenten bilden eine Anordnung in Schichten, wie in Abbildung 2.1 dargestellt. Die Struktur dieses Kapitels ist deshalb von unten nach oben gewählt: Damit sind die ersten drei zu analysierenden Komponenten die existierenden TCG-Erweiterungen des Bootloaders, der Linux-TDD und der TSS, gefolgt von der Analyse der drei Anwendungen TPM-Manager, den TPM-Tools und der OpenSSL-TPM-Engine.

2.1. GRUB und seine TC-Erweiterungen

Der Bootloader GRand Unified Bootloader[1] wurde ursprünglich von Erich Boleyn entwickelt und wird inzwischen vom GNU Projekt gepflegt. GRUB hat eine flexible Architektur und kann verschiedene Betriebssysteme (wie Linux, andere Unix-Derivate oder Microsoft Windows) booten. Er unterstützt weiterhin den Multiboot-Standard. Da GRUB unter der GNU General Public License (GPL) lizensiert ist, lässt er sich ideal um weitere Funktionen erweitern.

1 http://www.gnu.org/software/grub/

Wie fast alle anderen Bootloader besteht GRUB aus mehreren Teilen, die nacheinander ausgeführt werden. Der erste Teil (genannt stage1) befindet sich im Bootsektor. Die Aufgabe von stage1 besteht in der Konfiguration wichtiger Grundeinstellungen, wie beispielsweise den Adressierungsmodus des Bootmediums zu bestimmen, den nächsten Teil des GRUB zu Laden und zu der passenden Speicherstelle zu springen, die für den restlichen Bootvorgang verantwortlich ist. Der zweite Teil (genannt stage2) ist der Hauptteil des GRUB. Seine Aufgabe besteht in der Initialisierung des Systems und dem Laden des Betriebssystemkerns. GRUB bietet zusätzliche Funktionen an, wie z. B. die Unterstützung von Netzwerkkarten, um über ein Netzwerk zu booten.
Das Verhalten von GRUB wird durch die Konfigurationsdatei menu.lst bestimmt. Diese Datei enthält alle Parameter, die zum Booten eines Betriebssystems notwendig sind. Basierend auf dieser Datei stellt GRUB ein Bootmenü zur Verfügung, mit dem Benutzer Konfigurationen für den Bootvorgang auswählen und verschiedene Betriebssysteme (wie Linux oder Windows) booten können.

stage 1. `stage1` von GRUB enthält den Startcode des Bootloaders, der sich im Master Boot Record (MBR) befindet und durch das Basic Input Output System (BIOS) ausgeführt wird, nachdem es die wichtigsten Teile der Hardware initialisiert hat. Die Größe des Bootsektors ist auf 512 Bytes beschränkt, aber da im MBR zusätzlich noch die 64 Byte große Partitionstabelle enthalten ist, reduziert sich die effektive maximale Größe des Codes, der zum Starten verwendet werden kann, auf etwa 440 Bytes. Wie oben erwähnt ist die Hauptaufgabe von `stage1` das Laden der nächsten Teils des Bootloaders `stage2` und die anschließende Übergabe der Kontrolle an den geladenen Code. Da sich der zweite Teil auf verschiedenen Medien (wie z. B. der Festplatten oder einer Disketten) befinden kann, muss `stage1` die verschiedenen Adressierungsmodi der entsprechenden Medien unterstützen. Genauer gesagt enthält `stage1` drei verschiedene Adressierungsmodi, zwei für Festplatten (genannt Logical Block Addressing (LBA) und Cylinder Head Sector (CHS)), sowie einen für Disketten (eine angepasste Version von CHS). Nachdem `stage2` mit Hilfe eines Adressierungsmodus gefunden wurde, wird sein erster Sektor (wieder 512 Byte) in den Speicher geladen.

Schließlich springt `stage1` zu der Adresse des Speicherblocks, der gerade geladen wurde. Auf diese Weise wird die Kontrolle des Bootvorganges an den nächsten Teil übergeben.

stage 2. `stage2` ist der Hauptteil des GRUB. Seine Aufgabe ist das Laden des Betriebssystemkerns mit zusätzlichen Parametern. Da der Betriebssystemkern üblicherweise zu groß ist, um in den Speicherbereich des Real-Modus (Real-Mode) zu passen, wird `stage2` bereits im geschützten Modus (Protected-Mode) ausgeführt.

2.1.1. TrustedGRUB

Um einen vertrauenswürdigen Bootvorgang (trusted boot) zu realisieren, erweitert Trusted GRUB[2] den GRUB um Sicherheitsmechanismen, die durch das TPM zur Verfügung gestellt werden. TrustedGRUB benutzt die von der TCG spezifizierten Funktionen, um eine Vertrauenskette (chain of trust) weiterzuführen, die durch das Core Root of Trust for Measurement (CRTM) und das TCG-erweiterte BIOS begonnen wurden. Zusätzlich ermöglicht TrustedGRUB, Integritätsmessungen beliebiger durch den Benutzer der Plattform spezifizierter Dateien durchzuführen. Die wesentliche Motivation hinter der Entwicklung von TrustedGRUB bestand darin, dass das Turaya-Projekt[3] einen TCG-basierten, Multiboot-fähigen Bootloader benötigte. Faktisch kann TrustedGRUB jedoch alle Betriebssysteme booten, die durch den originalen GRUB unterstützt werden.

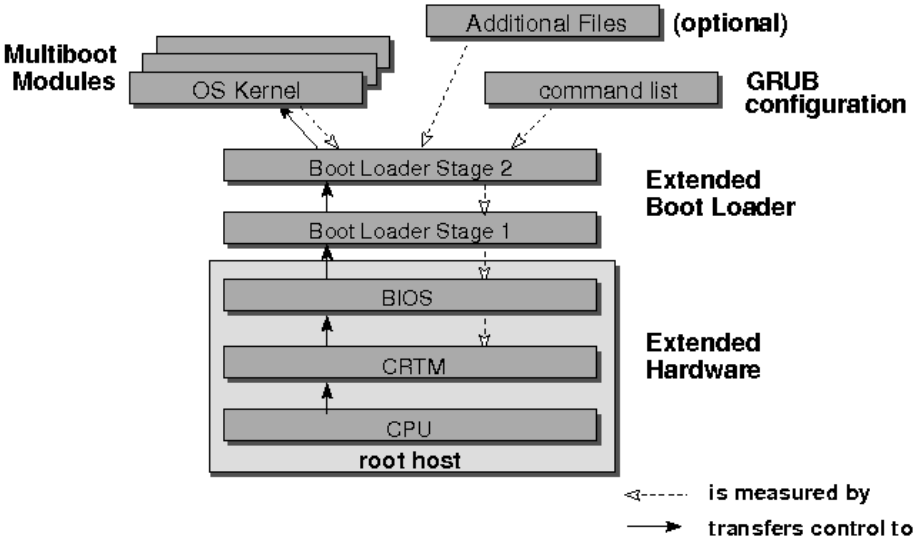


Abbildung 2.2.: Die am Bootprozess beteiligten Komponenten einer TCG-erweiterten PC-Plattform..

Abbildung 2.2 skizziert alle notwendigen Schritte zur Realisierung eines authentischen Bootvorganges durch TrustedGRUB und den Kontrollfluss zwischen dem BIOS, den GRUB-Ebenen und dem Betriebssystemkern. Der durch die TCG eingeführte Hauptunterschied zu einem normalen Bootvorgang ist, dass das CRTM das BIOS misst und das BIOS den Bootsektor. Darüberhinaus misst TrustedGRUB relevante Teile seiner Konfigurationsdatei, eine optionale Liste an Dateien und die zu ladenden Multiboot-Module.
TrustedGRUB erweitert `stage1` derart, dass es den ersten Sektor von `stage2` prüft. Ein

2 http://TrustedGRUB.sourceforge.net/
3 http://www.emscb.de/

Problem dabei ist die begrenzte Größe von `stage1`. Eine Lösung zu diesem Problem besteht in der aktuellen Version von TrustedGRUB darin, einige Adressierungsmodi (z. B. für den Diskettenzugriff) zu entfernen.
Um die Integrität des Bootprozesses zu garantieren, muss TrustedGRUB alle Eingaben prüfen, die von `stage2` verwendet werden, u. a. den Betriebssystemkern, weitere Multiboot-Module sowie Informationen zu deren Konfiguration. Die standardmäßigen Konfigurationseinstellungen des GRUB sind in einer Konfigurationsdatei `menu.lst` definiert, die von `stage2` geladen wird. Allerdings können Benutzer die Konfigurationsoptionen mit einem Kommandozeileneditor, der durch GRUB zur Verfügung gestellt wird, ändern oder erweitern. Die Konfigurationsoptionen aus der Konfigurationsdatei werden in einer Liste gespeichert und erst ausgeführt, wenn der Benutzer eine Konfiguration zum Booten ausgewählt hat. Verwendet ein Benutzer den Kommandozeileneditor, werden vom TrustedGRUB auch die eingegebenen Befehle gemessen und im TPM gespeichert.

2.1.1.1. Installation und Konfiguration

Um TrustedGRUB herunterzuladen und zu installieren, muss die Distribution von folgender Projektseite heruntergeladen werden: http://sourceforge.net/projects/TrustedGRUB
Danach muss das Archiv mit den in Listing 2.1 angegebenen Befehlen entpackt und übersetzt werden:

Listing 2.1: Konfiguration und Kompilierung von TrustedGRUB

```
# tar xzf TrustedGRUB-<ver >. tgz
# cd TrustedGRUB-<ver>
# . / build_tgrub . sh
# make install
```

Zusätzliche Parameter des Befehls `./build_tgrub.sh` können über die Option `--help` aufgelistet werden. Die Installation von TrustedGRUB kann manuell oder mit dem Tool `grub-install` erfolgen (siehe Listing 2.2).

Listing 2.2: Installation von TrustedGRUB in den MBR

```
 # cp stage1 / stage1 / boot / grub
 # cp stage2 / stage2 / boot / grub
 # . / grub / grub
 #   root (hdX,Y) // boot partition
 #   setup (hdX)  // hard disk to install TrustedGRUB on
 #   quit
Alternatively , use the grub install utility
located in the 'util ' subdirectory :
 # cd util
 # chmod a+x grub-install
```

```
# grub-install / dev / hdX
    # tar xzf TrustedGRUB-<ver >. tgz
    # cd TrustedGRUB-<ver>
    # . / build_tgrub . sh
    # make install
```

2.1.1.2. Zusätzliche Funktionalitäten

TrustedGRUB beinhaltet einige zusätzliche Funktionalitäten, die im Folgenden beschrieben werden:

Passwortüberprüfung. TrustedGRUB stellt eine Funktionalität zur Verfügung, das ein Benutzerpasswort abfragt, wenn der Parameter `--with-password-dialog` an ein Modul gehängt wird. In diesem Fall wird der Parameter `--with-password-dialog` durch den String `password=<yourpassword>` ersetzt, wobei <yourpassword> fü r das eingegebene Passwort steht. Diese Funktionalität liefert die Basis, um eine Authentifizierung vor dem eigentlichen Start des Betriebssystems (pre-boot) zu realisieren, wie sie durch einige Mikrokern-basierte Projekte gefordert wird.

Checkfile. TrustedGRUB kann die Integrität von beliebigen durch den Benutzer spezifizierten Dateien prüfen. Das ist durch die Einführung des neuen GRUB Befehls `checkfile` möglich. Der Benutzer teilt TrustedGRUB entweder über die Konfigurationsdatei `menu.lst` oder über die Kommandozeile mit, wo sich im Dateisystem ein sog. '`checkfile`' befindet.

Allgemein erlaubt die checkfile-Option die Überprüfung der Integrität aller Dateien in einem System. Im Gegensatz zur Messung der Multiboot-Module, die für das eigentliche Betriebssystem geladen werden, vergleicht TrustedGRUB bei der checkfile-Option die Messergebnisse mit Referenz-Hashwerten und unterbricht den Bootvorgang im Falle eines Fehlers. Dies dient dem Schutz vor manipulierten Komponenten und der daraus resultierenden Kompromittierung des IT-Systems.

```
checkfile (hd?,?)/somewhere/check.file
```

Es muss darauf geachtet werden, dass der Laufwerksparameter (hd?,?) und der Pfad richtig sind, andernfalls kann TrustedGRUB nicht booten. Das checkfile selbst enthält eine Liste von Tupeln beliebiger Länge (allerdings darf das checkfile nicht länger als 8192 Bytes sein) mit einer wie folgt definierten Syntax::

```
2647eeae7290c5a58dacb87347ba074de7e47bac (hd0,1)/etc/passwd
a97fbdba48d4a6340baff683941079dde56044e0 (hd0,1)/etc/shadow
d0fc1992068b660271af7bae7a1dea4258ef0b8b (hd0,1)/etc/group
bdf2a7d6132fe5ddf164b3bdae8764ab7433359a (hd0,1)/etc/fstab
```

Die erste Spalte enthält einen 40 Byte alphanumerischen Wert, der den SHA-1 Referenz-Hashwert

der zu messenden Datei repräsentiert (der Wert kann entweder mit `sha1sum` unter Linux oder mit dem Programm `create_sha1` erstellt werden, welches der TrustedGRUB-Distribution beiliegt). Danach folgt ein einzelnes Leerzeichen und die Pfad- und Dateinamenangaben der zu überprüfenden Datei.
Die Integrität aller im `checkfile` aufgelisteten Dateien wird beim Booten überprüft, indem der Referenz-Hashwert mit dem unmittelbar zuvor berechneten Hashwert verglichen wird. Ist ein Wert fehlerhaft wird eine Warnmeldung angezeigt. Der Benutzer hat dann die Möglichkeit den Startvorgang des möglicherweise kompromittierten Systems fortzusetzen oder den Bootvorgang abzubrechen. Diese Funktion bietet somit einen ersten Schritt in Richtung einer Secure-Boot Implementierung, wie in Abschnitt 1.1 definiert. Die Messwerte aller von der checkfile-Option überprüften Dateien werden in Platform Configuration Register (PCR) 13 des TPM gespeichert.
Die Integrität der `checkfile`-Datei selbst wird derzeit durch TrustedGRUB nicht überprüft. Ersetzt ein Angreifer eine Datei auf dem Dateisystem und den korrespondierenden Hashwert innerhalb der `checkfile`-Datei, so wird es beim Startvorgang zwar zu keinem Fehler kommen, da der gemessene Hashwert mit dem Referenzwert innerhalb der checkfile-Datei übereinstimmt. Allerdings wird der Hashwert der veränderten Datei ebenfalls in PCR 13 des TPMs gespeichert, sodass eine Manipulation der `checkfile`-Datei sich in jedem Falle in einem veränderten PCR 13 widerspiegelt. Um implizite Daten- oder Systemintegrität zu gewährleisten, kann man Daten an PCR 13 versiegeln - eine Manipulation wird dann automatisch entdeckt, sobald man versucht, die Daten wieder zu unsealen.

Verzeichnis	GRUB	TrustedGRUB
stage1	239	297
stage2	19380	22805
lib	1479	1482
util	1104	2220
grub	1103	1115

Tabelle 2.1.: Codezeilenvergleich der einzelnen Komponenten von GRUB und TrustedGRUB.

sha1 Dienstprogramm. Dieses neue GRUB-Dienstprogramm berechnet den SHA-1-Hashwert der angegebenen Datei und gibt das Ergebnis auf dem Bildschirm aus. Die Syntax von `sha1` lautet:

```
sha1 (hd?,?)/somewhere/hash/my/file
```

2.1.1.3. Qualität der Implementierung

Codekomplexität und Qualität. Der Hauptteil (ca 85%) des GRUB Bootloaders wie auch des TrustedGRUB ist in ANSI-C geschrieben (vgl. Tabelle 2.1), allerdings wurde an einigen Stellen Assembler benutzt, um den Code in den begrenzten Speicherplatz unterzubringen (etwa damit `stage1` in den 512-Byte großen MBR passt).

Aufgrund der Komplexität des Bootvorganges, der Größe des Dateisystems und der verschiedenen Bootmedien ist GRUB selbst ein sehr komplexes Programm. Um die Verwendbarkeit zu erweitern, bietet GRUB eine Menge Funktionalitäten, was abermals die Codegröße und damit auch die Komplexität erhöht.

Dokumentation und Unterstützung. Ein Handbuch ist in der TrustedGRUB-Distribution enthalten. Zusätzlich ist für TrustedGRUB[4] ein erweitertes Wiki und ein Issue-Tracking-System verfügbar, sowie die Mailinglisten auf der Sourceforge Projektseite[5] .

Limitierungen. Die folgenden Einschränkungen des TrustedGRUB wurden bisher erkannt:

- Mit einigen BIOS-Implementierungen funktioniert TrustedGRUB nicht, wenn das TPM im BIOS deaktiviert ist. Dies ist bisher beim IBM Thinkpad T41p aufgetreten.
- In einigen Notebooks kann TrustedGRUB die PCRs des TPM wegen der fehlenden BIOS-Funktionalität `TCG_PASS_THROUGH_TO_TPM` nicht erweitern. Dies ist kein grundsätzliches Problem von TrustedGRUB, sondern liegt an einem fehlenden oder fehlerhaften CRTM.
- Die Diskettenunterstützung ist zurzeit aufgrund des begrenzten Speicherplatzes von GRUBs `stage1` herausgenommen worden..

2.1.1.4. Zukünftige Arbeit

Die folgenden Verbesserungen von `stage2` des TrustedGRUBs sind geplant:

- Hinzufügen einer Konfigurationsoption, um das zu erweiternde PCR festzulegen.
- Unterstützung von Property-Based Attestation/Sealing, um Daten an einen öffentlichen Schlüssel anstatt einem binären Messwert zu binden [11].

2.1.2. GRUB-IMA

GRUB-IMA ist eine andere TC-Erweiterung des GRUB-Bootloaders und wurde von IBM für ihre Integrity Measurement Architecture (IMA) entwickelt. Wie auch TrustedGRUB misst GRUB-IMA alle GRUB-Komponenten und die zu ladenden Betriebssystemkomponenten. Nachdem die geladenen Dateien gemessen wurden, werden deren Hashwerte wie in Tabelle 2.2 beschrieben in die PCRs des TPM geschrieben. Zusätzlich werden die Messwerte in einer ACPI-Tabelle gespeichert. Tabelle 2.3 beschreibt die dabei verwendeten Ereignistypen.

PCR	Verwendung
4	MBR Information, `stage1` und `stage2`
5	ACTIONs
8	OS Komponenten (Kernel, Initrd, Module, Ergebnisse des Befehl `measure`)

Tabelle 2.2.: PCR Verwendung von GRUB-IMA.

4 http://projects.sirrix.com/trac/trustedgrub
5 http://sourceforge.net/projects/TrustedGRUB

Ereignis-Typ	Beschreibung
0x1005	SHA1-Wert der GRUB ACTION Nachricht
0x1105	SHA1-Wert der Kommandozeile
0x1205	SHA1-Wert des Kernel
0x1305	SHA1-Wert der Initrd
0x1305	SHA1-Wert der Module

Tabelle 2.2.: Ereignis-Typen von GRUB-IMA.

2.1.2.1. Installation und Konfiguration

GRUB-IMA kann von der TrouSerS-Projektseite auf Sourceforge.net[6] heruntergeladen werden. GRUB-IMA selbst steht nicht als eigenständiges Paket bereit, sondern wird lediglich als Änderung der normalen GRUB-Implementierung angeboten. Dies erfordert für die Installation ein Patchen der Originalquelle mit der bereitgestellten Quelltext-Differenz (Patch). Listing 2.3 beschreibt die erforderlichen Installationsschritte.

Listing 2.3: Installation von GRUB-IMA

```
      # rpm -ivh grub -0.97 -13. src . rpm
      # cd /usr/src/redhat/SPECS /
      # rpmbuild -bp grub.spec
      # cd ../BUILD/grub-0.97/
      # patch -p1 < grub-0.97-13-ima-1.1.0.0.patch

Build:
      # autoreconf --install --force
      #CflAGS=-Os  -g  -fno-strict-aliasing  -Wall  -Werror  \
        -Wno-shadow -Wno-unused -Wno-pointer-sign
      # export CflAGS
      #./configure  --sbindir=/sbin  --disable-auto-linux-mem-opt  \
        --enable-ima
      # make

Install GRUB-IMA (this step does not update the GRUB in MBR):
      # make install

final installation:
      # /sbin/grub-install install_device
```

6 http://sourceforge.net/projects/trousers/

2.1.2.2. Zusätzliche Funktionalitäten

GRUB-IMA bietet einige neue Befehle und Funktionen in der GRUB-Kommandozeile und in der GRUB-Konfigurationsdatei.

tpm. Innerhalb der GRUB-Kommandozeile kann der neue Befehl tpm wie in Listing 2.4 beschrieben benutzt werden.

Listing 2.4: tpm-Befehl in GRUB-IMA

```
tpm pcrs      zeigt die Werte der PCR-Register
tpm eventlog zeigt das vom BIOS und GRUB gemessene Eventlog
tpm test      testet die TCG-BIOS-Aufrufe von Int 1 AH
```

measure. Mit Hilfe des Befehls measure können beliebige Dateien gemessen und in die PCRs des TPMs gespeichert werden. Dies geschieht durch das Anfügen der Zeile `measure <filename>`, wie in Listing 2.5 gezeigt.

Listing 2.5: measure Befehl in GRUB-IMA

```
title CentOS (2.6.18-8.1.4.el5)
      root (hd0,0)
      measure /etc/selinux/config
      measure /etc/selinux/targeted/policy/policy.21
      kernel /boot/vmlinuz-2.6.18-8.1.4.el5 ro root=/dev/sda1 rhgb
      initrd /boot/initrd-2.6.18-8.1.4.el5.img
```

2.1.2.3. Qualität der Implementierung

Codekomplexität und Qualität. Der GRUB-IMA-Patch umfasst etwa 113 kB Code, der größte Teil davon besteht aus Assembler-Routinen. Die Quellcodeerweiterungen sind gut dokumentiert.

Dokumentation und Unterstützung. GRUB-IMA besitzt keine Dokumentation, Wiki oder Forum. Im Quelltextpaket ist lediglich eine kleine README-Datei enthalten. Unterstützung erhalten Entwickler und Nutzer über die Mailingliste des TrouSerS-Projektes.

Limitierungen. Da bei GRUB-IMA die SHA1-Berechnung durch den TPM-Chip durchgeführt wird, dauern die Messungen sehr lange. Außerdem kann der GRUB-IMA keine Mikrokerne (wie L4) ausführen, daher kann er nicht mit Architekturen wie z. B. Turaya verwendet werden.

2.1.3. OSLO

Der Open Secure LOader (OSLO) ist ein OSS Bootloader, der die SKINIT-Instruktionen der AMD64-Prozessorfamilie [2] verwendet. OSLO benötigt dazu ein TPM der Version 1.2 und kann damit ebenfalls zur Realisierung eines authentifizierten Bootvorgangs verwendet werden. Die Sicherheit basiert in diesem Fall auf einer dynamischen Vertrauenskette durch die Verwendung eines Dynamic Root of Trust for Measurement, wohingegen TrustedGRUB und GRUB-IMA auf einer statischen Vertrauenskette Static Root of Trust for Measurement basieren.
OSLO selbst wird durch einen Multibootloader (z. B. GRUB) gebootet. Danach initialisiert OSLO das Dynamic Root of Trust for Measurement, misst alle zu ladenden Dateien und speichert das Ergebnis in den dynamischen PCRs des TPM. Um mit dem TPM kommunizieren zu können, wird OSLO mit einem TPM Interface Specification (TIS)-Gerätetreiber ausgeliefert.

Die OSLO-Distribution besteht aus drei Teilen:

1. OSLO: Das Hauptprogramm.
2. Beirut: Ein Hilfsprogramm, welches die Kommandozeilen von Multiboot-Modulen misst.
3. Pamplona: Ein Hilfsprogramm, welches die Schritte von OSLO rückgängig macht. Beispielsweise entfernt es den Device-Schutz (DEV-Flag) und löscht die globalen Interruptflags. Es erlaubt somit, OSLO zu verwenden, aber ein unverändertes Betriebssystem in unsicherer Weise zu starten.

2.1.3.1. Installation und Konfiguration

OSLO kann unter http://os.inf.tu-dresden.de/~kauer/oslo/ heruntergeladen werden. Da OSLO eine dynamische Vertrauenskette verwendet, ist die Verwendung eines sicheren Bootloaders unnötig. Listing 2.6 zeigt, wie OSLO mit dem TrustedGRUB gestartet wird, so wie es im OpenTC-Projekt[7] Verwendung findet. Es lässt sich erkennen, dass der OSLO-Bootloader wie ein Multiboot-Modul behandelt wird.

2.1.3.2. Qualität der Implementierung

Codekomplexität und Qualität. OSLO besteht aus etwa 1000 Codezeilen. Die übersetzte Binärdatei hat eine Größe von etwa 4kB.

Dokumentation und Unterstützung. Innerhalb des Quelltextpaketes ist eine kleine Anleitung enthalten. Zusätzlich gibt es eine Mailingliste der Dresdener Operating System Group unter http://os.inf.tu-dresden.de/mailman/listinfo/l4-hackers/.

Limitierungen. Zur Zeit sind mindestens zwei Bugs bekannt: Die Device-Initialisierung fehlt und es gibt einen „type of memory“-Bug.

7 http://www.opentc.net

2.1.4. Bootstrapping

Die Bezeichnung *bootstrapping* wird oft durch das gebräuchlichere Wort *booting* (oder Booten) abgekürzt. Das Booten ist der notwendige Vorgang, um die grundlegende Hardware einer Plattform zu initialisieren (normalerweise wird das vom BIOS übernommen) sowie das Starten des Betriebssystems, was wiederum Aufgabe des Bootloaders ist. Dieses Kapitel beschreibt das dynamische Verhalten einer Plattform während des TCG-basierten Bootvorgangs.
Die Integrität einer Plattform kann nur gewährleistet werden, wenn die Vertrauenskette (chain of trust) nicht unterbrochen wird. Die Verankerung dieser Kette besteht aus der Kombination aus CRTM– in modernen PCs eine Erweiterung des BIOS– und dem TPM.
Eine funktionierende Vertrauenskette ist in mindestens einem der folgenden Szenarien erforderlich:

a) Jemand möchte die Integrität seiner eigenen Plattform absichern und auf vertrauliche Daten nur dann zugreifen können, falls das System unverändert im ordnungsgemäßen Zustand ist.
b) Jemand möchte die eigene Plattformkonfiguration einer externen Stelle beweisen. Ein mögliches Szenario ist ein Home-Offce-Mitarbeiter, der sich mit dem firmeninternen Netzwerk verbinden möchte, wobei die Firma den Zugang nur dann gewährt, wenn bestimmte Sicherheitsrichtlinien eingehalten werden und das IT-System des Mitarbeiters in einem ordnungsgemäßen Zustand ist.

Da die Vertrauenswürdigkeit der gesamten Plattform von jedem einzelnen Glied der Vertrauenskette abhängt, muss immer sichergestellt werden, dass diese nicht unterbrochen wird.

Erforderliche Schritte. Abbildung 2.3 zeigt die erforderlichen Schritte zum Aufbau einer gültigen Vertrauenskette bei einem TCG-basierten Bootvorgang nach dem Einschalten des IT-Systems.

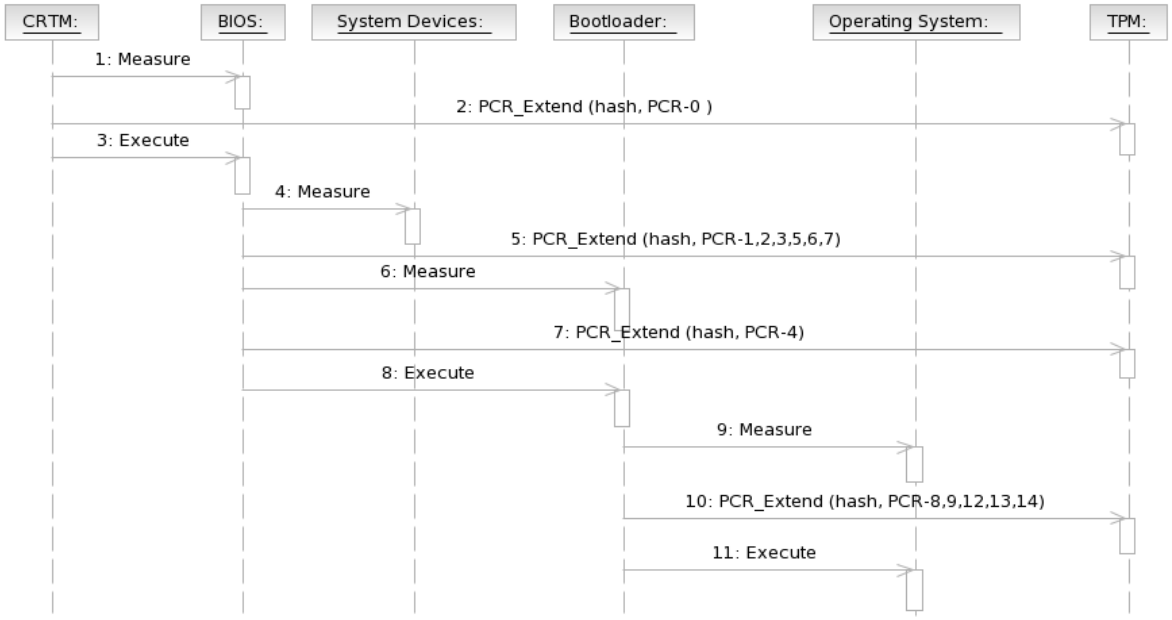


Abbildung 2.3.:Vertrauenskette eines TCG-basierten Bootloaders.

Listing 2.6: TrustedGRUB booted OSLO

```
# OTC - Proof of Concept prototype
# L4 boot.lst - for tGRUB 1.1.0 or newer versions
#
hiddenmenu
default=0
timeout=3
#
#OTCGRUBROOT, OTCL4 and LINUXROOT variables are set in the calling
#common menu.lst
#Other variables are set in the calling menu.lst specific for L4

title         OTC : booting L4 . . .
root          $ (OTCGRUBROOT)
kernel /boot/oslo/oslo
module /boot/oslo/beirut
module /boot/oslo/pamplona
module $(OTCSPEC)/bin/bootstrap
modaddr 0x02000000
module $(OTCSPEC)/bin/fiasco -nokdb -nowait
module $(OTCSPEC)/bin/sigma0 -v
module $(OTCSPEC)/bin/roottask -configfile -sigma0 task modname
       "bmodfs" attached modules task modname "compmgr" module
module $(OTCSPEC)/etc/roottask.config
module $(OTCSPEC)/bin/names
module $(OTCSPEC)/bin/events
module $(OTCSPEC)/bin/dm_phys -e -v --isa=0x800000
module $(OTCSPEC)/bin/simple_ts -e -t 200
module $(OTCSPEC)/bin/rtc -v
module $(OTCSPEC)/bin/l4io
module $(OTCSPEC)/bin/bmodfs
          module $(OTCSPEC)/bin/libld-l4 . s . so
          module $(OTCSPEC)/bin/libloader . s . so
          module $(OTCSPEC)/bin/vmlinuz-dom0
          module $(OTCSPEC)/bin/vmlinuz-domUT
          module $(OTCSPEC)/etc/$(DOM0)
          module $(OTCSPEC)/etc/$(DOMT)
          module $(OTCSPEC)/etc/$(DOMU)
          module $(OTCSPEC)/etc/$(DOMS)
          module /boot/preroot.img
          module $(OTCSPEC)/bin/loader
module $(OTCSPEC)/bin/mgui -r 1024 x768
module $(OTCSPEC)/bin/compmgr
module $(OTCSPEC)/etc/$(COMPMGRCONF)
module $(OTCSPEC)/bin/compmgrclientl4 -f BMODFS -t loader -l
       $(DOM0) $(COMPMGRGUESTS)
```

1. CRTM: Das CRTM misst das BIOS und schreibt das Ergebnis in das TPM.
2. BIOS: Das BIOS misst sämtliche auszuführende Firmware (etwa das Option-ROM der Grafikkarte) und den Bootloader (der im MBR des Bootmediums gespeichert ist). Das Ergebnis wird ebenfalls im TPM gespeichert.
3. Bootloader: Der Bootloader setzt die Vertrauenskette fort, indem er alle geladenen Sys temdateien misst und ebenfalls das Ergebnis in das TPM schreibt. Zusätzlich können optionale Dateien über die checkfile-Funktion mit in die Vertrauenskette aufgenommen werden. Details dazu stehen in Abschnitt 2.1.1.2.
4. Betriebssystemkern: An dieser Stelle muss das Betriebssystem dafür Sorge tragen, dass die Vertrauenskette fortgesetzt wird. Dies ist erforderlich, da sich das System während der Ausführung verändert.

Nach Abschluss des Bootvorgangs sollte das Betriebssystem geladen und jeder ausgeführte Code überprüft und im TPM gespeichert worden sein. Nun kann die Integrität der Plattform verifiziert oder einer anderen Stelle attestiert werden. Da die TCG-Definition von TrustedBoot direkt mit Abschluss des Bootvorganges endet, obliegt es dem nun laufenden Betriebssystems, die Vertrauenskette fortzusetzen. Mehrere Ansätze werden gerade untersucht und/oder implementiert. IBM beispielsweise hat eine IMA-Erweiterung für Linux[8] entwickelt (siehe auch Kapitel 4).

2.2. Linux TDD

Um unter Linux ein TPM zu verwenden, ist ein entsprechender TPM-Treiber notwendig. Wenn das TPM als Hardwaremodul im System integriert ist, muss ein spezifischer Treiber für das TPM geladen werden, um Zugang zur Hardware zu bekommen. Gemäß der TPM-Spezifikation wird diese Abstraktionsschicht TPM Device Driver genannt. Die folgenden Unterkapitel beschreiben den aktuellen Status der Implementierung der TPM-Unterstützung in Linux.

2.2.1. High-Level-Design

Generische TPM-Schnittstelle. Während der Anfänge der TPM-Spezifikation wurden keinerlei Anforderungen an das Design der TPM-Schnittstelle innerhalb des Betriebssystems gestellt[9] . Daher waren TPM-Hersteller gezwungen das Kommunikationsverhalten mit dem Chip selbst zu definieren. Darüberhinaus können verschiedene TPM-Anbieter verschiedene Kommunikationsschnittstellen anbieten, was es für Anwendungsentwickler sehr schwierig macht jeden TPM-Chip am Markt zu unterstützen. Um dieses Problem zu umgehen, entwickelte IBM eine vom TPM-Chip unabhängige Schnittstelle für Linux. Dies ermöglicht Anwendungsentwicklern TPM-Unterstützung unabhängig vom verwendeten Chip anzubieten.

8 http://domino.research.ibm.com/comm/research_people.nsf/pages/sailer.ima.html
9 Dies hat sich mittlerweile durch Einführung der TSS-Spezifikation [7] geändert.

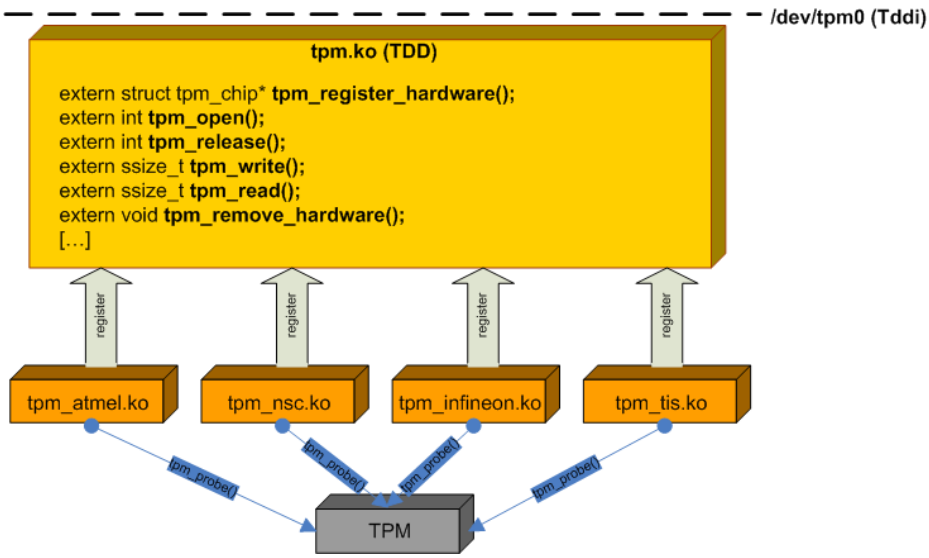


Abbildung 2.4.: Die TPM-Schnittstelle in Linux.

Die TPM-Schnittstelle ist inzwischen komplett in Linux integriert und erlaubt die Verwendung von mehr als einem TPM-Chip gleichzeitig. Um mit dem TPM zu kommunizieren, erlaubt Linux einen bidirektionalen Zugang zu der TPM-Schnittstelle über das `device`-Dateisystem. Der erste TPM-Chip wird in `/dev/tpm0` eingebunden, der zweite Chip bei `/dev/tpm1` und so weiter. Alle Befehle, die nach oder von z. B. `/dev/tpm0` schreiben bzw. lesen, müssen TPM-Befehle sein, wie in [24] und [27] definiert. Abbildung 2.4 zeigt die interne Struktur der Linux TDD-Architektur und seinen herstellerspezifischen Komponenten.

Zusätzlich zum device-Dateisystem, welches für die Kommunikation mit dem Chip verwendet wird, sind weitere TPM-Informationen über das `sys`-Dateisystem verfügbar unter `/sys/class/misc/tpmX`. Die `sys`-Dateisystemeinträge für das TPM bieten einen einfachen Weg, Informationen über den Chip zu erhalten, wie etwa den aktuellen Inhalt der PCRs, den öffentlichen Schlüssel des TPM oder verschiedene Fähigkeiten des Chips. Des Weiteren bietet es eine einheitliche Möglichkeit, laufende TPM-Operation zu beenden.

Listing 2.7 zeigt wie die aktuellen PCR-Werte ausgelesen werden, und Listing 2.8 zeigt wie eine laufende TPM-Operation abgebrochen werden kann.

Herstellerspezifische TPM-Treiber. Solange die äußere Sicht auf die TPM-Schnittstelle unabhängig von der aktuellen TPM-Chip-Implementierung ist, sind herstellerspezifische Treiber verantwortlich für die Kommunikation mit dem eigentlichen Chip. Diese Treiber sind von außen nicht zugänglich, da sie nur intern in Linux verwendet werden. Für TPMs der Version 1.1b existiert keine einheitliche Anforderung an den Kommunikationsprozess, daher werden für verschiedene TPM-Implementierungen jeweils eigene TPM-Treiber benötigt.

Die folgenden TPM 1.1b Treiber sind z. Z. unter Linux verfügbar (vgl. Abbildung 2.4).:

- Infineon TPM 1.1b
  `modprobe tpm_infineon`
- Atmel TPM 1.1b
  `modprobe tpm_atmel`
- NSC TPM 1.1b
  `modprobe tpm_nsc`

Generischer TPM 1.2 Treiber. Mit der Einführung der TPM-Spezifikation 1.2 [24] wurde eine standardisierte TPM-Kommunikationsschnittstelle (genannt TIS) spezifiziert [23]. Die zugehörige Implementierung der Spezifikation ist ebenfalls unter Linux verfügbar und unterstützt die meisten der verfügbaren TPM 1.2 Chips.

- TPM 1.2 TIS
  `modprobe tpm_tis`

Abbildung 2.5 zeigt, wie der TPM-Treiber in Linux ausgewählt wird.

Listing 2.7: Der Inhalt der PCRs via sysfs

```
# cat /sys/class/misc/tpm0/device/pcrs
PCR-00:BF D3 08 05 08 E7 98 D7 D2 CB 0E 15 A8 B8 74 DB 8F F8 CB A2
PCR-01:79 50 52 AB EB DB 2A 1B C7 73 0E 59 3D BA B4 6F CE 8C BE DE
PCR-02:EB B3 BA AE E7 57 4B B6 37 AA AB 67 0F 9A C1 BC EB 6F 80 F3
PCR-03:04 FD EC DD 50 1D AF 0F 62 4C 1F 99 60 12 CF 30 44 ff 46 10
PCR-04:CF C7 FA 94 66 30 B5 0C AF 55 44 F1 96 C1 04 4B 20 99 E6 98
PCR-05:04 FD EC DD 50 1D AF 0F 62 4C 1F 99 60 12 CF 30 44 ff 46 10
PCR-06:05 13 FA E7 AE 00 59 0B 4A 98 A3 E1 86 BD F0 C6 1C 2C 80 AA
PCR-07:04 FD EC DD 50 1D AF 0F 62 4C 1F 99 60 12 CF 30 44 ff 46 10
PCR-08:C1 EC 2A 49 F3 43 F5 A2 9E A6 A4 0C A6 82 36 11 DB 53 F9 5F
PCR-09:0B 47 F0 9C FA 1F 0E D8 E7 8D EA 5C 27 32 54 E9 CD BC 5F 80
PCR-10:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR-11:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR-12:3C D6 6E 89 17 4E CD F1 15 D9 BA D3 C7 5E 0F BF 9C A2 ED B0
PCR-13:6B C9 23 12 C7 32 36 90 ff 7D 5A D0 E0 D2 A6 00 29 9B 10 B0
PCR-14:39 21 82 94 E8 D5 0A B4 2E 3E 84 64 CC 41 3C 40 AB CA B5 0A
PCR-15:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Listing 2.8: Abbrechen der aktuellen TPM-Operation via sysfs

```
echo 1 > /sys/class/misc/tpm0/device/cancel
```

2.2.2. Qualität der Implementierung

Die Linux TPM-Schnittstelle sowie die implementierten Treiber funktionieren problemlos. Durch den aufwändigen Prozess der Kernel-Integration ist der Quellcode aktuell, gut gepflegt und besitzt eine hohe Implementierungsqualität, obwohl die Dokumentation der Quellen eher spärlich ist. Die TPM-TIS 1.2 Implementierung erfüllt die Anforderung der Spezifikation [23]. Gegenwärtig ist allerdings keine Anleitung verfügbar, wie das TPM innerhalb eines GNU/Linux Betriebssystems verwendet wird, allerdings ist Unterstützung über eine Mailingliste verfügbar[10].

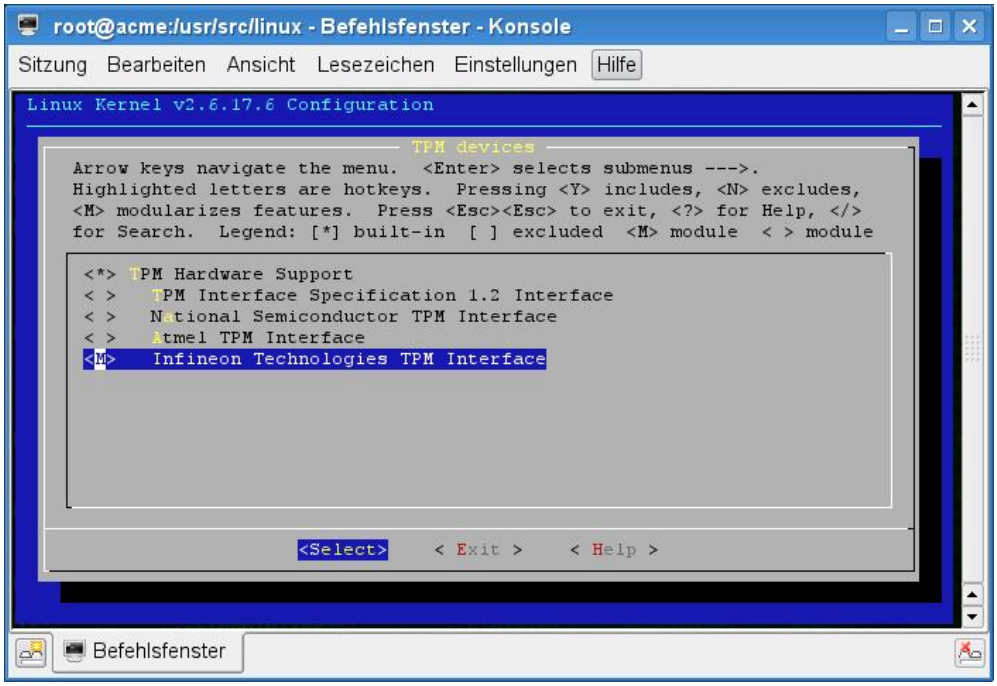

root@acme:/usr/src/linux - Befehlsfenster - Konsole
Sitzung Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
Linux Kernel v2.6.17.6 Configuration
TPM devices
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < > module
<*> TPM Hardware Support
< > TPM Interface Specification 1.2 Interface
< > National Semiconductor TPM Interface
< > Atmel TPM Interface
<M> Infineon Technologies TPM Interface
<Select> < Exit > < Help >
Befehlsfenster

Abbildung 2.5.: Auswahl der TPM-Gerätetreiber in Linux.

2.3. TrouSerS-TSS

Der TCG Software Stack stellt dem Betriebssystem und den Anwendungen eine Application Programming Interface (API) zur Verfügung, so dass diese die Funktionalitäten des TPM nutzen können. Der TSS erlaubt verschiedenen Anwendungen und verschiedenen Betriebssystemen, welche als virtuelle Maschinen auf einem Virtual Machine Monitor (VMM) laufen, das TPM zu verwenden, wenn sie den TSS-Spezifikationen genügen.
Die Hauptaufgabe des TSS ist den vielfältigen Zugang zum TPM zu verwalten, da das TPM nur beschränkte Kapazitäten besitzt und nur mit einem Klienten gleichzeitig kommunizieren kann.

10 https://lists.sourceforge.net/lists/listinfo/tpmdd-devel/

TrouSerS ist eine OSS TCG Software Stack-Implementierung, die seit 2004 von IBM gepflegt wird. Die aktuelle stabile Version implementiert TSS Version 1.1 [6] für TPMs der Version 1.1b. Seit die TCG ihre neueste Version der TSS-Spezifikation [7] herausgegeben hat, in der neue Funktionalitäten des TPM der Version 1.2 unterstützt werden, wird TrouSerS aktuell grundlegend überarbeitet. Zur Zeit sind noch nicht alle Funktionalitäten der Version 1.2 verfügbar. Der aktuelle Entwicklungsprozess kann jedoch auf [9] verfolgt werden.

2.3.1. High-Level-Design

Der TSS ist in vier Schichten aufgeteilt. Abbildung 2.6 zeigt eine konkrete TSS-Instanz.

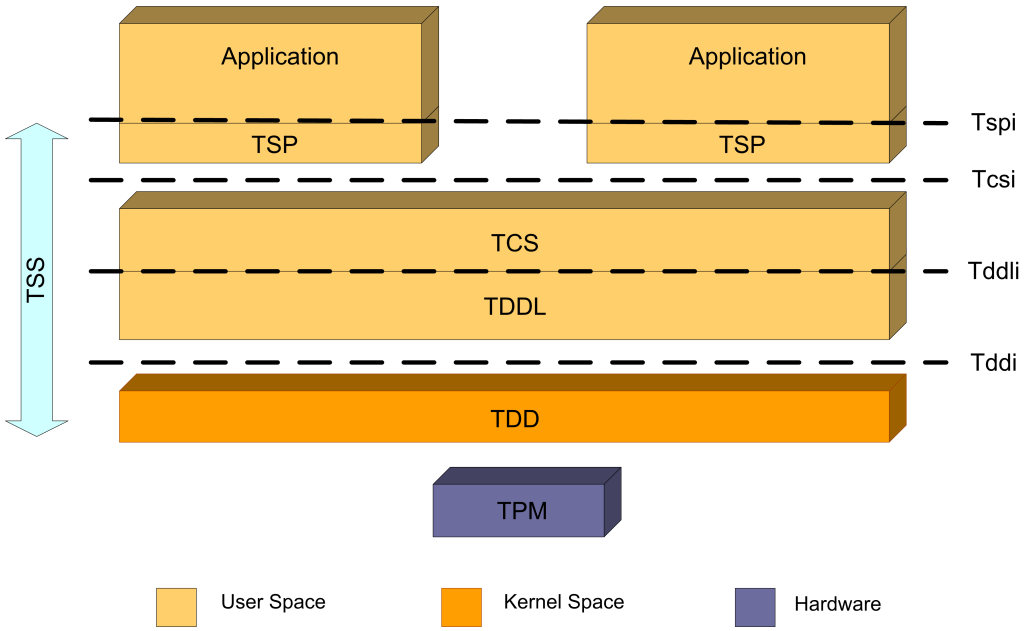


Abbildung 2.6.: Exemplarisches Objektmodell einer TSS-Instanz.

TSP und TSPI. Jede Anwendung, die TPM-Funktionalitäten nutzen will, muss dazu seinen TSS Service Provider (TSP) über seine Schnittstelle TSS Service Provider Interface (TSPI) benutzen. Jede Anwendung verwendet dazu ihren eigenen TSP, der innerhalb des Prozesses dieser Anwendung ausgeführt wird. Deshalb ist der TSP als Bibliothek realisiert. Der TSP abstrahiert High-Level-TCG-Funktionen, die etwa zum Hashen oder zur Signaturüberprüfung notwendig sind. Ein Teil des TSP ist der TSP Context Manager (TSPCM), welcher dynamische Handles zur effizienteren Verwendung von TSPs und Anwendungsressourcen zur Verfügung stellt. Der andere Teil sind die TSP Cryptographic Functions (TSPCF). Sie stellen die geschützten Funktionen des TPMs zur Verfügung. Das TSPI ist als C-Schnittstelle definiert.

TCS und TCSI. Der TSS Core Service (TCS) bedient alle TSPs einer Plattform. Da es pro TPM nur einen TCS gibt, ist seine Hauptaufgabe den mehrfachen Zugriff auf das TPM zu ermöglichen. Darüberhinaus liefert der TCS Operationen zum Speichern, Verwalten und Schützen von Schlüsseln und sensiblen Geheimnissen, die Bezug zur Plattform haben. Der TCS ist als User-Space-Dämon implementiert.

TDDL und TDDLI. Die TCG Device Driver Library (TDDL) mit Zugang über das TCG Device Driver Library Interface (TDDLI) bietet eine einheitliche Schnittstelle zu verschiedenen TPM-Treiberimplementierungen. Darüberhinaus stellt es den Übergang zwischen Kernel-Modus und Benutzer-Modus bereit[11]. Die TDDL stellt Funktionen bereit (etwa Open, Close, GetStatus), um die Verbindung zum TDD zu verwalten. Dazu liefert es Funktionen (GetCapability, SetCapability), um Attribute des TPM, TDD und TDDL auszulesen und zu setzen, sowie direkte Funktionen (Transmit, Cancel) zum Übertragen und Abbrechen von TPM-Kommandos.

TDD und TDDI. Der TPM Device Driver empfängt Nachrichten, die inhaltlich auf das spezifische Verhalten des TPM zugeschnitten sind. Er ist das einzige Modul, das direkt mit dem TPM kommunizieren kann. Er empfängt von der TDDL Byteströme und sendet diese an das TPM. Die Antworten des TPMs gehen dann direkt zurück an die TDDL. Obwohl die TSS-Spezifikation die Schnittstelle TDDI für den TDD erwähnt, ist sie in der aktuellen TSS-Spezifikation nicht definiert. Anders als die TDDL, die im User-Space implementiert ist, ist die TDD im Kernel-Space implementiert.

2.3.2. TSS und Linux TDD

Im Folgenden werden die Schnittstellen zwischen TSS und Linux-TDD beschrieben. Um Daten zwischen TrouSerS und TPM auszutauschen, macht TrouSerS Gebrauch von seiner eingebauten TDDL, die verantwortlich für die Kommunikation mit dem Linux TPM-Treiber ist (wie schon in Abschnitt 2.2 beschrieben). Die Hauptfunktionen der TDDL werden in der TSS 1.2 API spezifiziert.

Solange die Kommunikation zwischen der TSS und dem Linux-TDD innerhalb der TDDL abgewickelt wird, werden alle notwendigen Funktionen in einer Bibliothek namens `libtddl.a` implementiert (die nur von dem TCS-Dämon verwendet wird). Der Quelltext dieser Bibliothek befindet sich im TrouSerS-Paket unter `src/tddl/tddl.c`.

Nach dem Start von TrouSerS wird ein TCS-Dämon geladen, der zuerst die Existenz eines TPMs im System überprüft. Das passiert, indem nach verschiedenen Device-Nodes im `Device`-Dateisystem gesucht wird. Das bedeutet im Detail, dass die `Tddli_Open()`-Funktion nach `/dev/tpm`, `/dev/tpm0` und `/udev/tpm0` sucht. Falls erfolgreich, wird das Device durch TrouSerS geöffnet und ist danach für alle anderen Anwendungen blockiert.

11 Das jetzige TSS-Design geht von einem monolithischen Betriebssystem aus, in dem Gerätetreiber im Kernel-Modus laufen

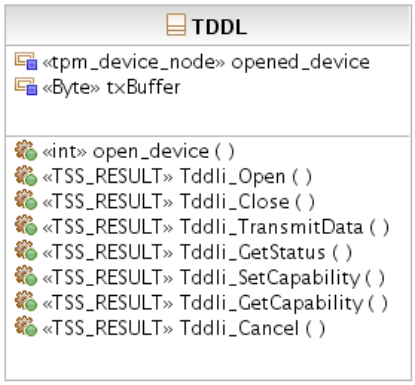


Abbildung 2.7.: Die Schnittstelle von der TCG Device Driver Library (TDDL).

Falls eine Anwendung den TSS verwendet, wird ein Bytepuffer erzeugt, der die TPM-Funktionsaufrufe einschließlich aller notwendigen Parameter enthält. Der `tcsd` kommuniziert dann mit dem TPM-Treiber unter Verwendung der `Tddli_TransmitData()`-Funktion.

Diese Funktion sendet Daten zu dem Device-Node des TPM im `Device`-Dateisystem und wartet auf die Antwort vom TPM-Treiber. Nachdem der TPM-Treiber die Daten an das TPM weitergeleitet hat, wird er die Antwort an das selbe Device-Node zurückliefern. Die Funktion `Tddli_TransmitData()` leitet die Antwort des TPM zurück an die aufrufende Anwendung. Bei Beendigung des `tcsd` entsperrt die Funktion `Tddli_Close()` den TPM-Treiber.
Abbildung 2.8 beschreibt exemplarisch die Kommunikationsfolge zwischen TrouSerS und TPM.

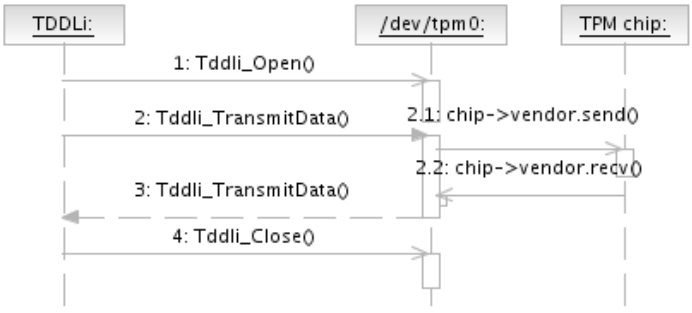


Abbildung 2.8.: Die Kommunikation zwischen TDDI und dem TPM.

2.3.3. Installation und Konfiguration

Installation. Die letzte stabile Version 0.3.1 von TrouSers wurde Anfang November 2007 veröffentlicht und kann unter http://sourceforge.net/projects/trousers/ heruntergeladen werden. Der Entwicklungszweig von TrouSerS wird mit CVS verwaltet. Um die aktuelle Version aus dem CVS herunterzuladen, müssen die folgenden Befehle ausgeführt werden:

```
# cvs -d:pserver:anonymous@trousers.cvs.sf.net:/cvsroot/trousers \
  login
# cvs -z3 \
  -d:pserver:anonymous@trousers.cvs.sf.net:/cvsroot/trousers co  \
  -P trousers
```

In dem neu erzeugten Ordner `trousers` kann der Code mit folgenden Befehlen kompiliert werden[12]:

```
# sh bootstrap.sh
# ./configure
# make
  as root
# make install
```

TSS 1.1	TSS 1.2
80% ANSI C	91% ANSI C
43,000 Codezeilen	56,000 Codezeilen

Tabelle 2.4.: Anzahl Codezeilen von TrouSerS version 1.1 und 1.2.

Nach einer erfolgreichen Installation von TrouSerS kann der TCS-Dämon `tcsd` durch die Ausführung von `startproc -u tss /usr/local/sbin/tcsd` gestartet werden.

Konfiguration. TrouSerS enthält eine Konfigurationsdatei unter `/etc/tcsd.conf`. Zusätzlich speichert TrouSerS seine TPM-Geheimnisse in der Datei `/var/lib/tpm/system.data`.

2.3.4. Qualität der Implementierung

Aufgrund der hohen Komplexität der Spezifikation ist auch die Implementierung eines TSS sehr anspruchsvoll. Der Quelltext von TrouSerS selbst ist in ANSI-C geschrieben und gut dokumentiert.

12 Hinweis: TrouSerS kann nur mit GCC 4.1 kompiliert werden

Testfälle. Die Entwickler von TrouSerS liefern etwa 450 Testfälle für ihre TSS-Implementierung der Version 1.1 und 1.2 in ihrem CVS. Um eine Testumgebung herunterzuladen, muss das folgende Repository ausgecheckt werden:

```
# cvs -d:pserver:anonymous@trousers.cvs.sf.net:/cvsroot/trousers \
  login
# cvs -d:pserver:anonymous@trousers.cvs.sf.net:/cvsroot/trousers \
  co -P testsuite
```

Dokumentation und Unterstützung. TrouSerS bietet für Entwickler und Benutzer verschiedene Mailinglisten. Zusätzlich existieren FAQs[13] auf der Projektseite. Innerhalb des Quellpaketes gibt es ein `doc`-Verzeichnis, welches tiefgehende Designdokumente der Implementierung enthält. Auch ein Handbuch für die meisten Kommandos ist verfügbar.

2.4. Die TrouSerS TPM-Tools

Das TrouSerS-Projekt bietet mit den TPM-Tools einige auf dem TSS aufbauende Tools zur Pflege und Diagnose eines TPM an. Zusätzlich enthält das Paket Kommandos, um die Möglichkeiten des `openCryptoki`-Projekts (eine PKCS#11 Implementierung für Linux) zu nutzen. Das Paket TPM-Tools steht unter der „Common Public“-Lizenz.

2.4.1. Installation und Konfiguration

Das TPM-Tools-Paket kann von der TrouSerS-Projektseite heruntergeladen werden unter: http://sourceforge.net/projects/trousers/. Nach Anklicken des Links ’Download TrouSerS’ erscheint der Link TPM-Tools. Um die Datei zu entpacken und zu installieren, müssen folgende Befehle ausgeführt werden:

```
# tar xzf tpm-tools-1.3.0.tar.gz
#  sh ./bootstap.sh
#  ./configure
#  make
als  root
#  make install
```

2.4.2. High-Level-Design

Dieser Abschnitt beschreibt die Interaktionen zwischen User-Space-Anwendungen und dem TSS am Beispiel der TPM-Tools.
Der Zugang zum TSS erfolgt über die TSP-Schnittstelle, die in der `libtspi.so`-Bibliothek implementiert ist. Obwohl diese Bibliothek in ANSI-C geschrieben wurde, besitzt sie ein objekt-

13 http://trousers.sourceforge.net/faq.html

orientiertes Design. Deshalb befassen sich alle Funktionen im TSPI mit einer oder mehreren Referenzen, die bestimmte Instanzen einer Klasse adressieren. Das TSPI, wie es durch `libtspi.so` bereitgestellt wird, definiert die in Abbildung 2.9 aufgezeigten Klassen.

Die wichtigste Klasse ist `Context`, die für die Verbindung zu dem TCS, der auf einer lokalen oder fremden Plattform ausgeführt wird, zuständig ist. Die Methoden dieser Klasse regeln Ressourcen, Speicher und andere Objekte. Ein anderer Schwerpunkt ist die Bereitstellung der Verbindung zum TCS und einer persistenten Speicherdatenbank. Die Interaktion zwischen TCS und TSP wird unter Linux durch Sockets realisiert. Das Socket, den der TCS-Dämon anspricht, hat Port-Nummer 30003. Die meisten Funktionen werden in der Klasse `TPM` definiert, sie repräsentiert den Besitzer eines TPM und bietet einige grundlegende Funktionen zur Kontrolle und zum Report. Die `Policy`-Klasse kann zur Konfiguration der Policy-Einstellungen verwendet werden. Die `Key`-Klasse bietet TCG-Schlüsselverwaltung und -funktionalität.

2.4.3. Funktionalität

Die aktuelle Version der TPM-Tools unterstützt siebzehn Kommandozeilen-Tools zur Pflege des TPMs. Folgende Funktionen sind enthalten:

- Besitzerverwaltung
- Endorsement Key-Verwaltung
- Konfigurationsverwaltung

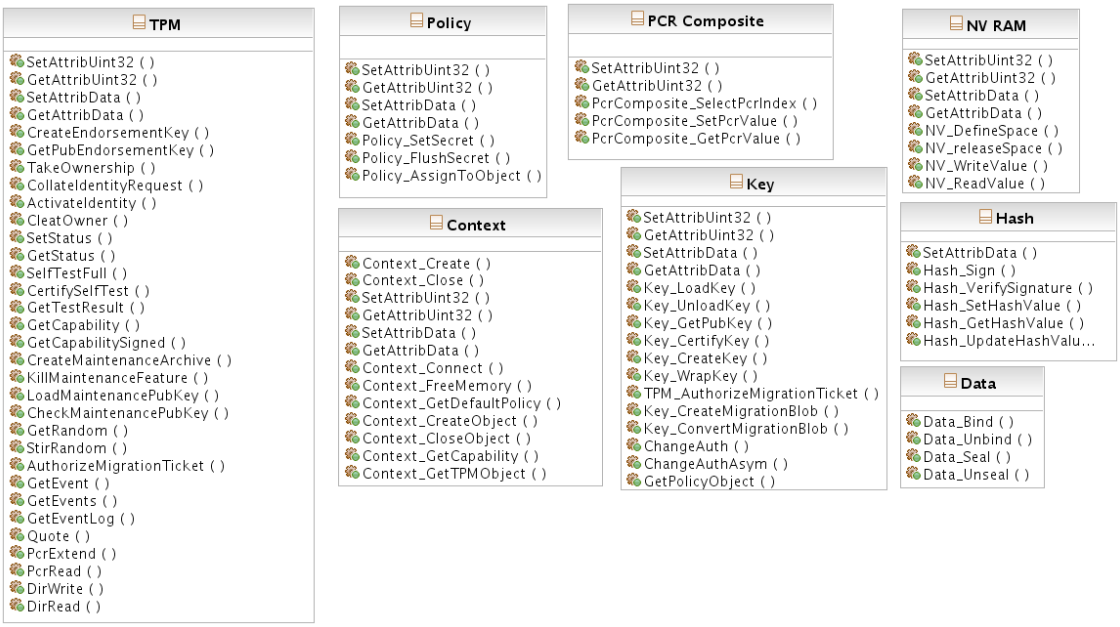


Abbildung 2.9.: Durch die TSP-Schnittstelle (TSPI) definierte Klassen.

Befehl	Erklärung
tpm_changeownerauth	Ändern der TPM-Benutzerauthorisierungsdaten
tpm_clear	TPM zurücksetzen
tpm_createek	Endorsement-Schlüsselpaar im TPM erzeugen
tpm_getpubek	Öffentlichen Teil des Endorsement-Schlüssel ausgeben
tpm_resetdalock	Gegenmaßnahmen gegen TPM-Angriffe zurücksetzen (erfordert Besitzerauthorisierungsdaten)
tpm_restrictpubek	Auslesen des öffentlichen Schlüssels verhindern
tpm_revokeek	Endorsement-Schlüssel des TPMs widerrufen
tpm_sealdata	Daten mit Hilfe des TPM an eine Konfiguration versiegeln
tpm_selftest	Selbsttest des TPM
tpm_setactive	TPM-Aktiv-Status ändern
tpm_setclearable	TPM-Status löschbar machen
tpm_setenable	TPM Ein- / Ausschalten
tpm_setoperatorauth	Besitzerauthorisierungs-Informationen setzen
tpm_setownable	TPM zur Besitzübernahme („Take Ownership“) freischalten
tpm_setpresence	TPM-Status zur physischen Anwesenheit ändern
tpm_takeownership	Besitz des TPM ergreifen („Take Ownership“)
tpm_version	TPM Version und Herstellerinformationen ausgeben

Tabelle 2.5.: Befehle der TPM-Tools.

Tabelle 2.5 zeigt die möglichen Befehle der TPM-Tools auf. Um beispielsweise Besitzer eines TPM zu werden, muss folgendes eingeben werden:

```
# tpm_takeownership
```

In Tabelle 2.6 werden die fünf Befehle des PKCS#11-Data-Managements aufgelistet.

Befehl	Erklärung
tpmtoken_import	X.509-Zertifikat und/oder RSA Schlüsselpaar importieren
tpmtoken_init	TPM PKCS#11 Datenspeicher des Benutzers initialisieren
tpmtoken_objects	Objekte im Datenspeicher anzeigen
tpmtoken_protect	Ver- und Entschlüsselung von Daten mit Hilfe symmetrischer Schlüssel aus dem Datenspeicher
tpmtoken_setpasswd	Assoziierte Passwörter des TPM PKCS#11 Datenspeicher ändern

Tabelle 2.6.: Befehle für PKCS#11-Data-Management

2.4.4. Sealing mit den TPM-Tools

Um das dynamische Verhalten zwischen TPM-Tools und TSS aufzuzeigen, beschreiben wir exemplarisch, wie eine beliebige Datei an PCR 12 und 14 mit den TPM-Tools versiegelt werden kann. Dazu wird das Kommando `tpm_sealdata` mit folgenden Parametern aufgerufen:

```
tpm_sealdata –i file.to.seal –o sealed.file –p 12 –p 14.
```

Das Sequenzdiagramm in Abbildung 2.10 zeigt den Datenfluss sowie alle relevanten Unterfunktionen. Nach dem Aufruf der Funktion `tpm_sealdata` (Schritt 1) bezieht `tpm_sealdata` Zufallszahlen vom TPM (Schritt 1.7). Dazu ruft die `tpmGetRandom`-Funktion die Methode `Tspi_TPM_GetRandom()` der Klasse `TPM` auf. Dann setzt `tpm_sealdata` die SRK Policy (Schritte 1.9 bis 1.13) unter Verwendung der Klassen `Policy` und `Context`. In den nächsten Funktionen (Schritte 1.15 bis 1.21) wird ein RSA-Schlüssel-Objekt durch das TPM erzeugt. Danach wird ein RSA-Schlüssel generiert und geladen (Schritte 1.23 bis 1.25). Mit den nächsten Funktionen wird ein verschlüsseltes Datenobjekt hergestellt, in dem die verschlüsselte Version des symmetrischen Schlüssels enthalten ist (Schritte 1.27 bis 1.33). Die weiteren Funktionen verschlüsseln die gegebenen Daten und versiegeln sie an den symmetrischen Schlüssel. Es ist möglich, diesen Befehl mit verschiedenen Parametern aufzurufen. Beispielsweise bindet der Aufruf `tpm_sealdata –p <PCR>` Daten an ein oder mehrere PCR. Im erfolgreichen Fall werden die verschlüsselten Daten in die durch `–o` spezifizierte Datei geschrieben oder an `stdout` ausgegeben. Listing 2.9 zeigt die Ausgabe der erfolgreichen Versiegelungs-Operation:

Derzeit wird TrouSerS ohne ein `unseal`-Werkzeug ausgeliefert. Um ein Unseal von Daten durchzuführen, die durch `tpm_sealdata` versiegelt wurden, muss eine Funktion der durch TrouSerS mitgelieferten `tpm-Unseal`-Familie verwendet werden. Wenn alle Bedingungen korrekt sind, also der SRK nicht verändert wurde und die PCRs, die zur Zeit des Versiegelns spezifiziert wurden, den entsprechenden Wert haben, wird der ursprüngliche Inhalt der Datei in einem bereitgestellten Datenpuffer zurückgegeben. Eine einfache Implementierung könnte wie folgt aussehen:

```
if ((res=tpmUnsealfile(argv[1],&data,&size))==0){<do something>}.
```

Anhang A beschreibt eine einfache Implementierung eines Kommandos zum Unsealen mit dem Namen `tpm_unseal`.

Listing 2.9: Ergebnis von tpm_sealdata

```
-----BEGIN TSS-----
-----TSS KEY-----
AQEAAAARAAAABAEAAAABAAMAAQAAAAwAAAgAAAAAAgAAAAAAAAAAAAAABAM8d0Lh+
W79J99Al5KIXe25xIra0fd8yDMcXjmf/JeCK+c3V4oH5EnvdLfwVe/IbDfrnBEtQ
cn2OnweXNwIDpWNmjwbvrZhQCw1k4pIa45uGM/qo0Vfm/WbJ/JbXR2aXTcE+Xd/V
Oij+3XHHtddZjkPUtQUiHsj5pQA+gVkppNISMmrR/WbzIIUewC/GjH+DHjHm2+aW
Niu77ncvG6pbpFrPuxuJwYeqe2u8xsbXYdC5Vx4ou7+8JSq2nyoWDE0zL7lVrzBB
EWJWU+6dVX2fGMp1slK3V1XP4hzZmkI/1pcvLe06Pbmdc+Kk+u6Mi44Qqnwpn6eu
AImFxom3I8gg/nMAAAEAmyKLfMbJh5ZXLcif4dEltRt2kI+oI/3aS5D+SyFK74CO
1ohACDATJyQf0I+KmA52X9tG35CIwzQFBhAZsNf7LJ2RLNk25t6SWdV5BF3KWa2y
Y7kMk9qlgyEK852nwQ3hwKf6kdJJRc6cq1jUjqlBCFHFZ+/pWYsrskxenA0WssV8
fG3MeXZxau+5CfDTGoTrpTCJmzcxtQSCQu7puzcLvo4mxyuldIi1ImcXGJwBBHgA
L9AAVeDMqeajz9YuiFqV0AuqgqVQrKSiAc0c0tUvtlJ/YeCtzvlUf8HrFZrGmWlp
rsu2Q5u/OxbXVkPrZG7HiFqz90hhrsPA78dD7The4A==
-----ENC KEY-----
Symmetric Key : AES-256-CBC
AQEAAAAAACwAAgBQhBbi5yRMhzTZsLVa4Xk68CSuR1+EFuLnJEyHNNmwtVrheTrw
JK5HXwAAAQBRE91fMOjAS5U0RD+K3EA9aZQ2cLw3J4BO+65mjKo4Bm8KJgw6rFeq
TbuXl0qCdOq/env0Eb5V3+/sx3wOzBN6bIVUEbvk9DxcXA9LiJy/X4vR2wdVtaEJ
N49+MWTeEma95QV/gzvM4bUNEsiQdxg+fzbgGG6aoOjDbND1/ATBXCRxYJ+zxky9
O/ RrPrZZV6qsV1mAr2ERMdxs1AHp5CAMMfRWQdwC0hmD0RkciqPW7WE6xdhxp2rp
L8NAUgeVDwe2m90a4FhQFYflRB0vknfn1O5t/JtsGLD1lQQ8DZQVZhhEyfiRRbgE
wENOfacFeteAE+YcxPfYzOpCLIIoSwFM
-----ENC DAT-----
vWe+mpv7TVAsXs89t68eeA==
-----END TSS-----
```

2.4.5. Qualität der Implementierung

Codekomplexität. IBM hat seine Implementierung in ANSI-C geschrieben und der Code besteht aus etwa 5400 Zeilen. Der umfangreichste Teil ist die Bibliothek. Tabelle 2.7 zeigt die genaue Datenverteilung.

Struktur und Stil. Der Code ist gut strukturiert. Die meisten Funktionen sind kurz und eindeutig, meistens mit nicht mehr als 50 Zeilen und weniger als vier Parametern. Die Behandlung von Ausnahmen (*exceptions*) verwendet allerdings *goto*-Sprünge.

Abbildung 2.10.: Datenfluss nach dem Aufruf von tpm_sealdata.

High-Level-Design. Alle Programme werden in einer Kommandozeile ausgeführt. Sie benutzen dieselbe Bibliothek.

Dokumentation und Unterstützung. Der Code ist an wenigen Stellen dokumentiert. Zu jedem Befehl existiert ein Handbuch, welches auch unter http://trousers.sourceforge.net/man.html abgerufen werden kann. Support-Anfragen können unter http://trousers.sourceforge.net/ gestellt werden.

Limitierungen. Die aktuelle Veröffentlichung der TPM-Tools enthält Unterstützung des `TSS_WELL_KNOWN_SECRET` für den Storage Root Key (SRK). Das gilt leider nicht für `tpm_sealdata` und `tpmUnsealFile`, weshalb der Quelltext gepatched werden muss, um den Interoperabilitätstest in Kapitel 3.1 durchzuführen. Ein Fehlerbericht inkl. Korrekturvorschläge wurde an die Entwickler gesandt. Die verwendeten Patches stehen in Anhang B.

Verzeichnis	Anzahl Codezeilen
lib	1944
src_data_mgmt	1520
src_tpm_mgmt	2089
include	242
src_cmds	231

Tabelle 2.7.: Codekomplexität der TPM-Tools.

2.5. TPM-Manager

Das Ziel des „TPM-Manager“-Projektes [14, 21] ist die Realisierung einer OSS TPM-Management-Software mit einer benutzerfreundlichen graphischen Bedienungsoberfläche. Der TPM-Manager kann momentan unter Linux verwendet werden, wobei spätere Versionen auch als isolierte Anwendung auf einem Sicherheitskern wie Turaya[14] oder OpenTC[15] benutzbar sein sollen. Das jetzige Release des TPM-Manager wurde von der Sirrix AG[16] und der Ruhr-Universität Bochum[17] entwickelt und unter der GNU GPL-Lizenz Version 2 veröffentlicht.

2.5.1. Installation und Konfiguration

Der TPM-Manager ist auf einer Projektseite bei Sourceforge http://sourceforge.net/projects/tpmmanager/ verfügbar, wo man auch die aktuelle Version 0.4 des Quellcode herunterladen kann. Die Distribution verwendet `automake/autoconf` und kann wie in Listing 2.10

14 http://www.emscb.com/
15 http://www.opentc.net/
16 http://www.sirrix.com/
17 http://www.trust.rub.de/

beschrieben konfiguriert und installiert werden.

Listing 2.10: Konfiguration und Kompilierung des TPM-Manager

```
# tar xzf tpmmanager-0.4.tar.gz
# cd tpmmanager-0.4
# ./configure
# make
# make install
```

Da der TPM-Manager auf einigen KDE-Widgets basiert, müssen passende Header und Bibliotheken von KDE und Qt installiert sein. Der TPM-Manager wurde erfolgreich unter KDE 3.5 und Qt3 kompiliert.

2.5.2. High-Level-Design

Wie man in Abbildung 2.11 sehen kann, besteht das Design des TPM-Managers aus zwei Schichten, der GUI-Schicht und der MicroTSS-Schicht.
Während die GUI-Schicht Benutzerschnittstellen bereitstellt, die auf Widgets des KDE[18]- und des Qt[19]-Widget-Frameworks basiert, bietet die MicroTSS-Schicht eine einfache, kleine und objekt-orientierte Schnittstelle zu den Funktionen des TPM.

Obwohl die jetzige Implementierung des MicroTSS auf dem TrouSerS TSS (vgl. Abschnitt 2.3) aufsetzt, erlaubt die Abstraktion der MicroTSS-Schicht sowohl die Unterstützung alternativer TSS-Implementierungen durch dritte als auch die direkte Implementierung der benötigten Funktionen. Letzteres ist besonders wichtig, falls der TPM-Manager in einer sicherheitskritischen Umgebung, z. B. als Teil eines Sicherheitskerns, benutzt werden soll.

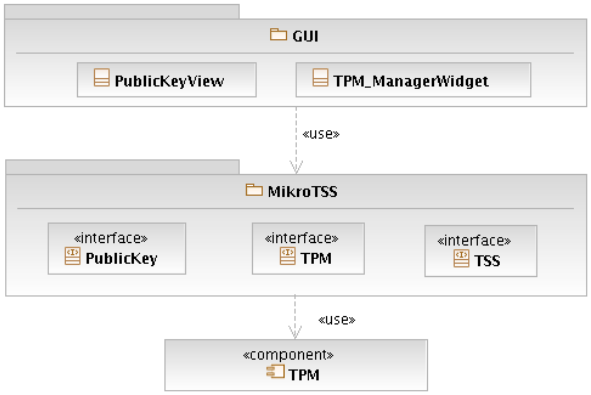


Abbildung 2.11.: Das High-Level-Design des TPM-Managers.

18 http://www.kde.org/
19 http://www.troll.no/

Die GUI-Schicht. Die GUI-Schicht des TPM-Managers stellt die Schnittstelle zum Benutzer bereit und verwendet die MicroTSS-Schicht, um Zugang zu den TPM-Funktionen zu erhalten. Die Logik der User-Interface-Schicht ist durch die Qt-basierte Klasse `TPM_ManagerWidget` realisiert. Ihre Basisklasse `TPM_ManagerWidgetBase` wurde mit dem Qt-Designer erstellt, Trolltech's Werkzeug zum Designen und Erstellen von GUIs. Der Qt-Designer erlaubt die Erweiterung neuer Funktionalitäten ohne grossen Aufwand. Abbildung 2.12 zeigt ein Bildschirmfoto des TPM-Managers.

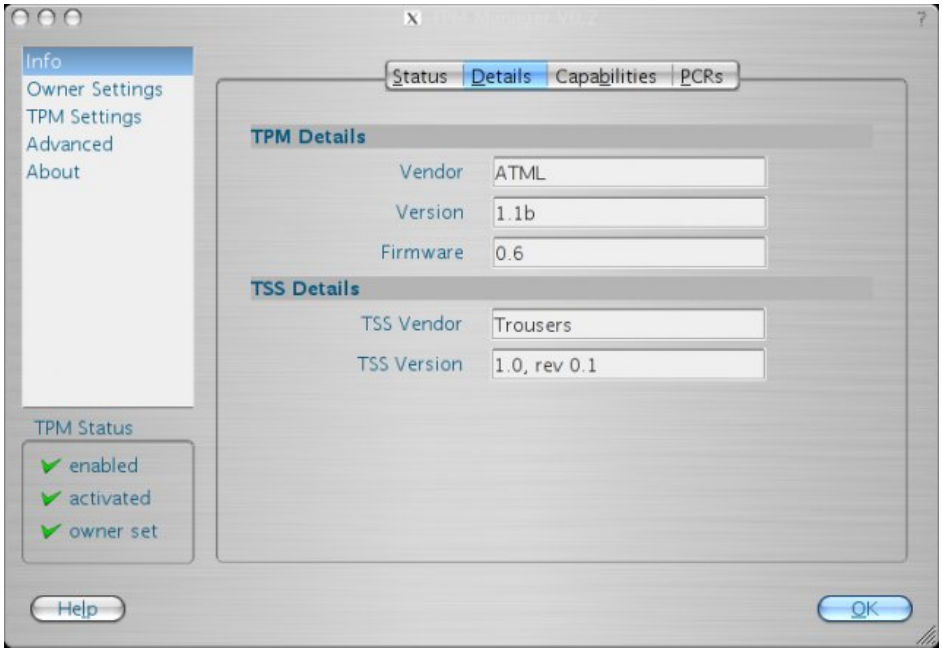


Abbildung 2.12.: Bildschirmfoto des TPM-Managers.

Die MicroTSS-Schicht. Die MicroTSS-Schicht stellt eine abstrakte Schnittstelle bereit, um Zugang zum TPM zu erlangen und verbirgt dabei Implementierungsdetails. Abbildung 2.13 zeigt das Unified Modeling Language (UML)-Modell der öffentlichen Schnittstelle und die Komponenten, die der TPM-Manager und das TSS Service Provider Interface bereitstellt. Die Hauptkomponente ist die Klasse `TPM`, welche die TPM-Mangement-Funktionen implementiert. Die Klasse `TPM` wird durch die Klasse `TSS` erstellt, die den Zugang zur verwendeten TSS-Implementierung kapselt. Beispielsweise prüft die Klasse `TSS` die Anwesenheit des TPM-Treibers, bevor ein Objekt vom Typ `TPM` erzeugt wird. Die Klasse `PublicKey` liefert Informationen, Typen und Attribute von kryptographischer Verschlüsselung und Testschlüsseln, die von der TSS verwaltet werden. Die MicroTSS-Schnittstelle beinhaltet bisher nur eine kleine Anzahl an Funktionen der Version 1.2 der TSS-Spezifikation, der Hauptteil der Implementierung basiert auf der TrouSerS Version für TPMs der Version 1.1b.

Die jetzige Implementierung der Klasse `TPM` selbst verwendet die `TSPI`-Schnittstelle, die durch die Bibliothek `libtspi.so` von TrouSerS bereitgestellt wird. Obwohl die `TSP`-Schnittstelle einige Funktionen beinhaltet, zeigt Abbildung 2.13 nur diejenigen Funktionen, die nötig sind, um den Anwendungsfall „Take Ownership“, wie in Abschnitt 2.5.3 beschrieben, zu realisieren.

Abbildung 2.13.: Schnittstellen, die durch den TPM-Manager und den MicroTSS bereitgestellt und benutzt werden.

2.5.3. „Take Ownership“ mit dem TPM-Manager

Um das dynamische Verhalten des TPM-Manager zu beschreiben, wird im Folgenden exemplarisch auf den Anwendungsfall „Take Ownership“ eingegangen. Abbildung 2.14 beschreibt mittels eines Sequenzdiagramms den Nachrichten- und Kontrollfluss beim Aufruf der „Take Ownership“-Funktion des TPM-Manager. Nachdem der Anwender den entsprechenden Knopf in dem TPM-Manager-Dialog drückt, wird die Methode `slotTakeOwnership()` der Klasse `TPM_ManagerWidget` aufgerufen (Schritt 1). Dabei werden zwei Passwort-Dialoge erstellt, um das TPM-Owner-Passwort und das SRK-Passwort einzugeben (Schritt 1.1 und 1.3). Danach wird der MicroTSS, genauer die Methode `takeOwnership()` der Klasse TPM aufgerufen (Schritt 1.5). Diese Methode verbirgt die Komplexität der Schnittstelle von `libtspi.so` durch den Aufruf von TSPI-Funktionen zum Setzen des Owner-Passwortes, des SRK-Passwortes und schließlich die Besitzübernahme des TPM (Schritt 1.5.1 bis 1.5.11). Abschließend informiert ein Dialog den Anwender über das Ergebnis der „Take Ownership“-Operation (Schritt 1.7).

2.5.4. Qualität der Implementierung

Codekomplexität und Design. Der TPM-Manager ist - basierend auf einem UML-Modell - mit Hilfe des „Rational Software Architects“, einer von IBM entwickelten Entwicklungsumgebung in C++ implementiert. Das GUI-Paket des TPM-Managers enthält etwa 3200 Zeilen Quellcode, wobei das MicroTSS-Paket etwa 857 Codezeilen umfasst.

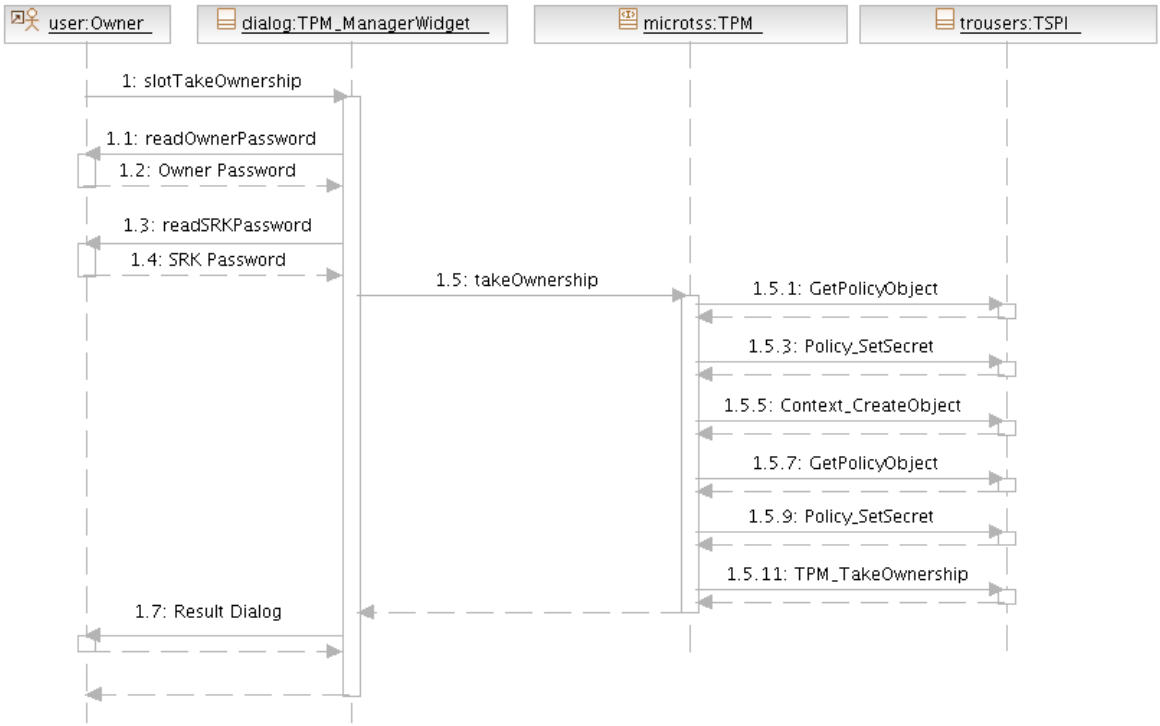


Abbildung 2.14.: Kontrollfluss nach Aufruf der „Take Ownership“-Funktion des TPM-Managers.

Dokumentation und Unterstützung. Der technische Report [21], der in der TPM-Manager-Distribution enthalten ist, enthält genaue Beschreibungen der realisierten Anwendungsfälle und des High-Level-Designs. Allerdings sind Implementierungsdetails sowie ein Benutzerhandbuch nicht vollständig vorhanden. Mailinglisten, ein „bug tracker“ und ein Diskussionsforum sind auf der Sourceforge-Projektseite verfügbar.

Limitierungen. Zur Zeit werden die wichtigsten TPM-Management-Funktionen unterstützt. Noch nicht funktionsfähig sind:

- Erstellung und Löschung des Endorsement Key wird nicht unterstützt

2.5.5. Zukünftige Arbeiten

Das nächste Release des TPM-Managers soll in erster Linie kleinere Fehler beseitigen. Langfristig sind die folgenden Erweiterungen geplant:

- Portierung zur TSS Version 1.2 und Implementierung weiterer Anwendungsfälle
- Portierung zu einer anderen Widget-Bibliothek mit geringerer Komplexität, etwa dem „Fast Light Toolkit“[20] (FLTK).

2.6. OpenSSL-TPM-Engine

OpenSSL [13] ist ein OSS Toolkit, welches das „Secure Sockets Layer“-Protokoll (SSL v2,v3) und das „Transport Layer Security“-Protokoll (TLS v1) implementiert, einschließlich der notwendigen kryptographischen Operationen. Das Toolkit basiert auf der SSLeay-Bibliothek [28] und wurde entwickelt von Eric A. Young und Tim J. Hudson. Es ist lizensiert unter einer Apache-ähnlichen Lizenz, was bedeutet, dass es für kommerzielle und nicht kommerzielle Zwecke verwendet werden kann.
Die OpenSSL-Bibliothek unterstützt die Erstellung, Manipulation und Verwendung von kryptographischen Modulen in der Form von *engine*-Objekten, d. h. Container für Implementierungen von kryptographischen Algorithmen. Solange diese Objekte dynamisch geladen werden können, kann OpenSSL um alternative Softwareimplementierungen oder Hardware-Module wie Smartcards oder ein TPM erweitert werden. Die kryptographische Funktionalität, die durch eine OpenSSL-Engine bereitgestellt werden kann, enthält folgende Abstraktionen:

- Alternative RSA, (EC)DSA, und (EC)DH Implementierungen
- Symmetrische Verschlüsselungsalgorithmen
- Hash-Algorithmen
- Laden und Speichern von kryptographischen Schlüsseln und Zertifikaten

2.6.1. OpenSSL-Engine-Schnittstelle

Alle Funktionen, die zur Implementierung und zur Verwaltung von OpenSSL-Engines benötigt werden, sind in der Datei `openssl/engine.h` definiert. Abbildung 2.15 zeigt die Funktionen, die von einer OpenSSL-Engine verwendet werden, um neue kryptographische Grundfunktionen zu unterstützen.
Implementierungen der OpenSSL-Schnittstelle können optional auch eine Teilmenge von diesen Funktionen definieren. In diesem Fall werden die voreingestellten Implementierungen der anderen Operationen verwendet.

20 http://www.fltk.org/

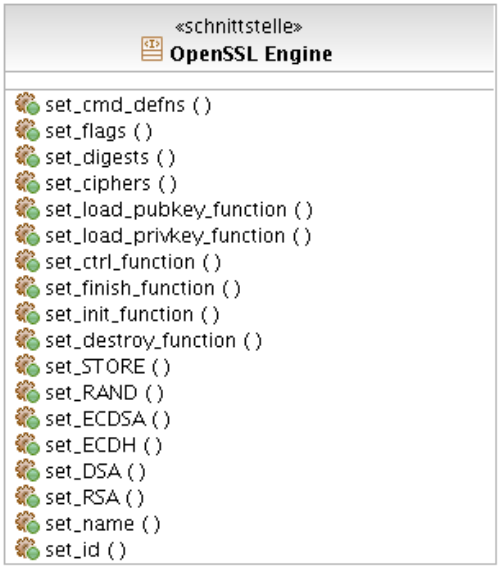


Abbildung 2.15.: Die generische OpenSSL-Engine-Schnittstelle.

2.6.2. OpenSSL-Engine-Aktivierung

Unglücklicherweise sind das jetzige Design und die Implementierung der OpenSSL-Engine-Unterstützung nicht transparent zur Anwendungsschicht. Stattdessen müssen Anwendungen OpenSSL-Funktionen aufrufen, um verfügbare OpenSSL-Engines explizit zu initialisieren und zu registrieren. Zwei Herangehensweisen sind dabei möglich: Entweder ist die entsprechende Engine im Quelltext der Anwendung direkt aktiviert, wie in Listing 2.11 gezeigt, oder die Anwendung muss die Unterstützung von OpenSSL durch folgenden Eintrag aktivieren: `#define OPENSSL LOAD CONF.`

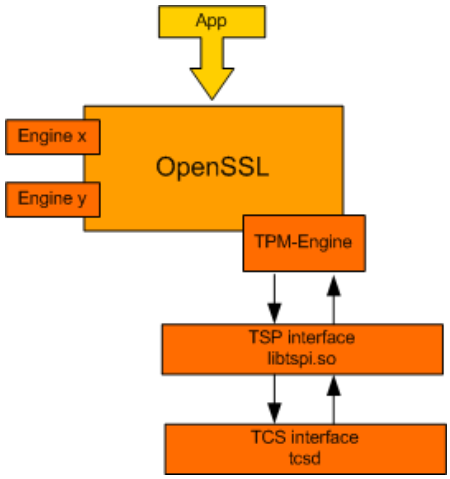


Abbildung 2.16.: Interaktion zwischen OpenSSL und TSS.

2.6.3. Die TPM-Engine

Die OpenSSL-TPM-Engine ist Teil des TrouSerS-Projektes [9]. Die Engine verwendet die OpenSSL-Engine-API, so dass durch ein TPM kryptographische Operationen durchgeführt werden können.
Wie in Abbildung 2.16 gezeigt, verbindet die TPM-Engine OpenSSL mit einem TPM mittels der TSS. Um das TPM aufzurufen, verwendet die TPM-Engine die gemeinsam genutzte Bibliothek `libtspi.so`.

Listing 2.11: Aufruf einer OpenSSL-Engine

```
/* Make the "dynamic" ENGINE available */
void ENGINE_load_dynamic ( void ) ;
/* Make the CryptoSwift hardware acceleration support available */
void ENGINE_load_cswift ( void ) ;
/* Make support for nCipher's "CHIL" hardware available */
void ENGINE_load_chil ( void ) ;
...
/*    Make    ALL    ENGINE    implementations    bundled    with
OpenSSL available */
void ENGINE_load_builtin_engines ( void ) ;
```

2.6.3.1. Konfiguration und Installation

Die Distribution der OpenSSL-TPM-Engine `openssl_tpm_engine-0.x.y.tar.gz` kann von der TrouSerS-Seite unter http://trousers.sourceforge.net/ heruntergeladen und mit den in Listing 2.12 gegebenen Befehlen konfiguriert und installiert werden. Es muss

sichergestellt werden, dass der Pfad zur bestehenden Installation von OpenSSL korrekt ist, andernfalls kann OpenSSL die TPM-Engine nicht finden. Um die TPM-Engine zu verwenden, muss ein Code-Fragment, wie in Listing 2.13 definiert, eingefügt werden.

Listing 2.12: Konfiguration and Installation von der OpenSSL-TPM-Engine

```
# tar xzf openssl_tpm_engine-0.x.y.tar.gz
# cd openssl_tpm_engine-0.x.y
# ./configure --with-openssl=/path/to/openssl
# make
as root
# make install
```

Listing 2.13: Aufruf der OpenSSL-TPM-Engine

```
/* Make all engine implementations bundled with
OpenSSL available */

void ENGINE_load_builtin_engines ( void ) ;

/* Select TPM engine */
ENGINE *e = ENGINE_by_id ( "tpm" ) ;

/* Initialize TPM engine */
ENGINE_init ( e ) ;

/* Make t h e TPM engine the default */
ENGINE_set_default_RSA ( e ) ;
ENGINE_set_default_RAND ( e ) ;
```

2.6.3.2. Qualität der Implementierung

Codekomplexität. Die OpenSSL-TPM-Engine ist in ANSI-C implementiert und besteht aus etwa 1825 Zeilen Quellcode. Die bereitgestellte Testumgebung umfasst etwa 174 Code-Zeilen.

Tests. Die OpenSSL-TPM-Engine wird mit einer kleinen Testumgebung geliefert, die vier Tests beinhaltet:

1. Laden eines TPM Schlüssels aus einer Datei mittels des Engine-Load-Befehls.
2. Testlauf mit dem geladenen TPM-Schlüssel.
3. Aufruf von `stir_random` durch die RAND-Schnittstelle.
4. Testdurchlauf der Autorisierungsdaten durch die TPM-Engine.

Dokumentation und Unterstützung. Der Quelltext der TPM-Engine ist spärlich dokumentiert. Darüberhinaus gibt es nur ein kleines README und eine beispielhafte Konfigurationsdatei, die zwei Anwendungsfälle beschreibt. Unterstützung bekommt man durch eine Mailingliste im TrouSerS-Projekt.

Probleme und fehlende Funktionalitäten. Unglücklicherweise fehlt eine passende Dokumentation, wie die OpenSSL-Engine verwendet werden kann. Bisher erfolgten keine Reaktionen seitens einer Anfrage auf der OpenSSL-Mailingliste. Deshalb war es nicht möglich, die TPM-Engine aus einem anderen Programm heraus transparent zu benutzen oder die OpenSSL-Konfigurationsdatei zu verwenden.

2.6.4. TPM-basierte „Transport Layer Security“ mit der OpenSSL-TPM-Engine

Um das dynamische Verhalten der OpenSSL-TPM-Engine zu erläutern, beschreibt dieser Abschnitt, wie eine TPM-geschützte Transportschicht erzeugt werden kann. Da das TPM die Generierung von RSA-Schlüsseln ermöglicht, wird das nötige Vorgehen erklärt, um einen TPM-geschützten RSA-Schlüssel innerhalb von OpenSSL zu verwenden. Zusätzlich zeigt dieser Abschnitt, wie ein Zertifikat von einem TPM-Schlüssel zum Zwecke der Verifikation und der Verwendung innerhalb eines Webservers erzeugt wird.
Die geforderten Schritte zum Aufbau eines verschlüsselten Kanals unter Verwendung eines TPM-geschützten Schlüssels sind:

1. Installation der OpenSSL-TPM-Engine wie in Abschnitt 2.6.3 beschrieben.
2. Erstellung eines Signaturschlüssels mittels:
   `create_tpm_key <keyfilename>`
3. Erstellung eines selbstsignierten Zertifikats:
   ```
   openssl req -keyform engine -engine tpm -key <keyname> -new \
   -x509 -days 365 -out <certfilename>
   ```
4. Starten des OpenSSL-Test-Servers:
   ```
   openssl s_server -cert <certname> -www -accept 4433 \
   -keyform  engine -engine tpm -key <keyname>
   ```

Abbildung 2.17 beschreibt den Kontrollfluss von Schritt 1 bis Schritt 4 auf. Um das Ergebnis zu testen, kann ein Webbrowser geöffnet werden, der auf den passenden Port zeigt, etwa durch Eingabe von `konqueror https://localhost:4433`. Der Browser sollte eine OpenSSL-Infoseite zeigen, unter Verwendung eines verschlüsselten Kommunikationskanals.

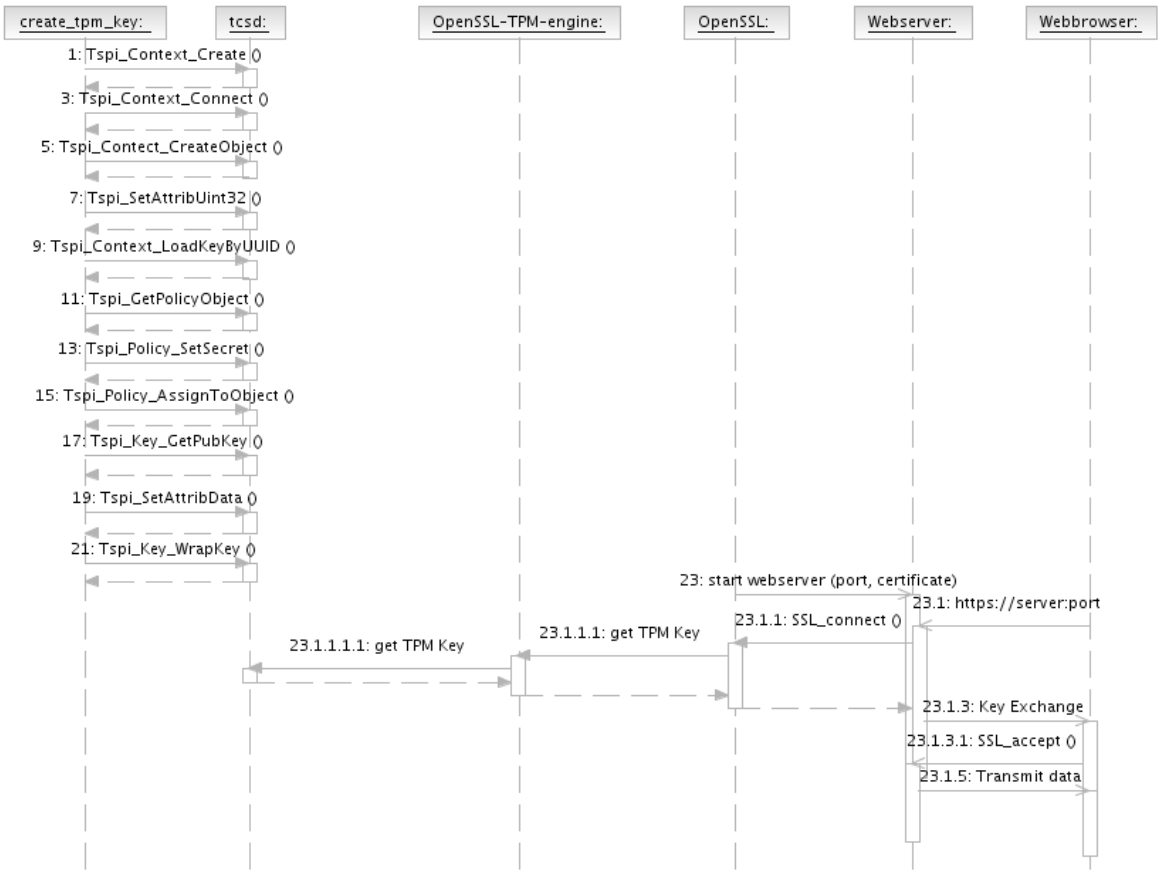


Abbildung 2.17.: Allgemeine (informell) Sequenz der TPM-Engine mit OpenSSL.

2.6.5. Zukünftige Entwicklungen

Gemäß der OpenSSL-Dokumentation werden die API der Engine und die interne Architektur gerade überarbeitet. Weitere Unterstützung für transparentes Laden von dynamischen *Engines* ist geplant. Dadurch würde die Entwicklung und Nutzung von OpenSSL-Engines erleichtert, die somit unabhängig von der verwendeten OpenSSL-Bibliothek und/oder OpenSSL-basierten Anwendungen sind.

3. Einhaltung der Spezifikationen und Interoperabilität

Das Ziel dieses Kapitels ist zu untersuchen, inwiefern die in Kapitel 2 analysierten Komponenten den TCG-Spezifikationen genügen (*compliance*) und ob eine Zusammenarbeit der Komponenten untereinander möglich ist (*interoperability*). Zum Test der Interoperabilität werden, basierend auf drei verschiedenen OSS Betriebssystemarchitekturen, nämlich SE-Linux, dem Xen-Hypervisor und dem Turaya-Sicherheitskern, bestimmte TC-Testfälle evaluiert.

In Abschnitt 3.1 werden die OSS Komponenten - soweit möglich - daraufhin untersucht, inwieweit sie den bestehenden Spezifikationen entsprechen. Die Testumgebungen für die Interoperabilitätstests werden in Abschnitt 3.2 erklärt, während die Testergebnisse in Abschnitt 3.3 diskutiert werden.

3.1. Compliance-Analyse

3.1.1. Bootloader Compliance

Die Compliance-Analyse der beschriebenen Bootloader basiert auf zwei verschiedenen Messungsansätzen. TrustedGRUB und GRUB-IMA verwenden das Static Root of Trust for Measurement, OSLO hingegen verwendet das Dynamic Root of Trust for Measurement für seine Messungen. Während die benötigten Funktionalitäten für das Static Root of Trust for Measurement durch das BIOS bereitgestellt werden, verwendet das Dynamic Root of Trust for Measurement anbieterspezifische Instruktionen. In beiden Ansätzen werden die Messungen in den PCRs des TPM gespeichert, welche in der ’TCG PC Client Specification’ [22, 26] definiert sind und in Tabelle 3.1 und Tabelle 3.2 aufgezeigt werden.

3.1.1.1. TrustedGRUB

Wie in Abschnitt 2.1.1 beschrieben, erweitert TrustedGRUB die PCRs mit den Messungen der zu ladenden Module. Dazu ist TrustedGRUB auf die passende BIOS-Unterstützung einer TCG-gesicherten Plattform angewiesen. TrustedGRUB verwendet die in Tabelle 3.3 definierten PCRs.

Da die ’TCG PC Client Specification’ [22, 26] die Verwendung von PCRs 0-7 definiert, aber nicht explizit die Verwendung der höheren Register 8-15, erfüllt die aktuelle TrustedGRUB-Implementierung die Spezifikation der TCG.

Um die Messungen des TrustedGRUB nachzuprüfen, bietet die Distribution des TrustedGRUB das Werkzeug `verify_pcr`. Beispielsweise kann damit der resultierende PCR-Wert nach der Messung der Datei `/boot/vmlinuz` mit dem folgenden Befehl in einer Shell überprüft werden:

```
$ verify_pcr NULL /boot/vmlinuz
-> Result for PCR:
39 21 82 94 e8 d5 0a b4 2e 3e 84 64 cc 41 3c 40 ab ca b5 0a
```

PCR Index	Alias	pcrReset
0 - 15	**Statisches RTM**	**0**
0	CRTM, BIOS und Plattform-Erweiterungen	0
1	Plattform-Konfiguration	0
2	Option ROM Code	0
3	Option ROM Konfiguration und Daten	0
4	IPL Code	0
5	IPL Code Konfiguration und Daten	0
6	Zustandswechsel und Ereignisse	0
7	Reserviert	0
8 - 15	Unspezifiziert	0

Tabelle 3.1.: Verwendung der S-RTM-PCRs, wie durch die TCG definiert.

PCR Index	Alias	pcrReset
16 - 23	**Dynamisches RTM**	**1**
16	Debug	1
17	Locality 4	1
18	Locality 3	1
19	Locality 2	1
20	Locality 1	1
21	T/OS kontrolliert	1
22	T/OS kontrolliert	1
23	Anwendungsspezifisch	1

Tabelle 3.2.: Verwendung der D-RTM-PCRs, wie durch die TCG definiert.

Das `verify_pcr`-Werkzeug funktioniert auch mit dem Inhalt von PCR13, der alle beliebigen durch die checkfile-Methode geladenen Dateien enthält. Wie vorher beschrieben, enthält das checkfile eine Liste an zu messenden Dateien, während das Ergebnis zur Erweiterung eines PCR verwendet wird.

```
$ verify_pcr NULL /etc/passwd /etc/shadow [...]
-> Result for PCR:
6b c9 23 12 c7 32 36 90 ff 7d 5a d0 e0 d2 a6 00 29 9b 10 b0
```

Dies zeigt, dass (i) die Messungen richtig berechnet und (ii) sie auch korrekt in das PCR gespeichert wurden.

PCR	Verwendung
4	MBR Information und `stage1`
8	`stage2` Part 1
9	`stage2` Part 2
12	Kommandozeilenargumente
13	`checkfile` Messungen
14	Kernel und Module

Tabelle 3.3.: PCR Verwendung von TrustedGRUB.

3.1.1.2. GRUB-IMA

Die GRUB-IMA Messungen werden in die in Tabelle 3.4 definierten PCRs gespeichert. Die Verwendung der PCRs ist gemäß der 'TCG PC Client Specification' [22, 26]. Um die Messungen zu verifizieren, kann die ACPI-PCR-Erweiterungs-Logdatei unter Linux wie folgt eingesehen werden:

```
# mount -t securityfs /sys/kernel/security
# cat /sys/kernel/security/tpm0/ascii_bios_measurements
# cat /sys/kernel/security/tpm0/binary_bios_measurements
```

PCR	Verwendung
4	GRUB `stage1` (MBR) bis `stage2` werden gemessen
5	ACTIONs
8	OS Komponenten (Kernel, Initrd, Module, Ergebnisse des Befehl `measure`)

Tabelle 3.4.: PCR Verwendung von GRUB-IMA.

Die `binary_bios_measurements` enthalten die genaue ACPI-Tabelle (worin das Integrity-Measurement-Log (IML) enthalten ist). Das `ascii_bios_measurements` ist hiervon das menschlich lesbare Format. Listing 3.1 zeigt ein Beispiel der BIOS-Logdatei.

Listing 3.1: Beispielausgabe von ascii_bios_measurements

```
0 98a1f3605524e874d068ac05be199cb0cb6c5e04 08 [S-CRTM Version]
0 ae81f0c566a34dbcefe204ade2b863622355daf4 01 [POST CODE]
0 d62689661aa4d0c5c93f4ed92b8fa7d592a45cd3 01 [POST CODE]
0 dd261ca7511a7daf9e16cb572318e8e5fbd22963 01 [POST CODE]
<snip>
4 c1e25c3f6b0dc78d57296aa2870ca6f782ccf80f 05 [Calling INT 19h]
4 38f30a0a967fcf2bfee1e3b2971de540115048c8 05 [Returned INT 19h]
4 cfa550a3b0fa6f8be76b8cf2d68be28355e57e2f 05 [Booting BCV Device 80h]
4 d6cf7a4dc273b2750b9791a2ba760c6fc6dc14a4 0d [IPL ]
5 1c47735c2cd4a0e2f33ca1730f85038f18d45e30 0e [IPL Partition Data ]
4 f3e37129afac034af2edba9d46f3571b5113ca0a 0d [IPL]
4 2929efe0b6d3582c0b23ea287b278b85d5cf33ee 0d [IPL]
4 5dbe0ae807ae0ecdb3e489603457d19c445c9c46 0d [IPL]
4 f6019d76ba7d20659d28d225a3305c9485533f95 06 []
4 8cdc27ec545eda33fbba1e8b8dae4da5c7206972 04 [GRUB Event Separator]
5 8cdc27ec545eda33fbba1e8b8dae4da5c7206972 04 [GRUB Event Separator]
5 cb9384a8c98849f08e21da4842fac3fd7cd9a1a0 0e [IPL Partition Data]
5 e32ba4121c9624586564659738014 33b537ec85e 05 [Password Prot. using MD5]
5 d63d12ced978aca120bfe6ee7683e394c2ffaef0 05 [Boot Seq. User Interv.]
5 486a8ca8bb98a87b3c0e5cb0e375b4640bad6bb9 1105 []
8 0b2adebe5ff3ee1f1f8a77e3cdba7f82948516ab 1205 []
8 1287bdda3d76ebed9b705d51c6f03b8f06cbf5a1 1305 []
5 2431ed60130faeaf3a045f21963f71cacd46a029 04 [OS Event Separator]
8 2431ed60130faeaf3a045f21963f71cacd46a029 04 [OS Event Separator]
8 fac33a1fc0ad42c07d00322d64c23f67567f334a 1005 []
```

In diesem Beispiel:

```
8 0b2adebe5ff3ee1f1f8a77e3cdba7f82948516ab 1205 [] is linux kernel
8 1287bdda3d76ebed9b705d51c6f03b8f06cbf5a1 1305 [] is initrd image
```

Also kann man diese Messung unter der Verwendung von `sha1sum` überprüfen.

```
# sha1sum /boot/vmlinuz-2.6.18-8ima.el5
0b2adebe5ff3ee1f1f8a77e3cdba7f82948516ab /boot/vmlinuz-2.6.18-8ima.el5

# sha1sum /boot/initrd-2.6.18-8ima.el5.img
1287bdda3d76ebed9b705d51c6f03b8f06cbf5a1 /boot/initrd-2.6.18-8ima.el5.img
```

3.1.1.3. OSLO

Die Messungen des OSLO Loaders werden in die in Tabelle 3.5 definierten PCRs gespeichert.

PCR	Verwendung
17	Messungen
19	Kommandozeilen-Messwerte

Tabelle 3.5.: PCR Verwendung von OSLO.

OSLO - als sicherer Bootloader - darf laut Spezifikation die beiden Localities 2 und 3 verwenden, da diese beiden Localities für sichere Betriebssysteme und sichere Initialisierungssoftware vorgesehen wurden. Das definierte PCR für Locality 2 ist PCR 19, daher entspricht OSLOs Wahl, die Messungen der Kommandozeilenparameter in PCR 19 zu erweitern, den Spezifikationen. Allerdings misst OSLO Binärdateien und speichert das Resultat in PCR 17, welches aber für Locality 4 vorgesehen ist und nur im Zusammenhang mit spezieller Initialisierungshardware verwendet werden sollte.

3.1.2. TrouSerS TSS

Compliance-Tests. IBM selbst betrachtet seine TSS 1.1 Implementierung als vollständig. Allerdings ist eine Untersuchung der Version 1.1 von TrouSerS kein Teil dieser Studie.

Verfolgt man den Ablauf der TSS 1.2 Implementierung, dann geben die Entwickler genauere Informationen über der gegenwärtigen Stand auf ihrer Projektseite unter http://trousers.sf.net/trousers12support.html. Da sich TrouSerS 1.2 aktuell noch in Entwicklung befindet, ist es gerade nicht möglich, etwas zur TSS-Implementierung zu sagen.

Laut der TrouSerS-Webseite arbeitet IBM gerade an 42 TSS-bezogenen Funktionalitäten, die sich in vier Bereiche einteilen lassen:

1. Eine vollständige Menge an Features im CVS: 19
2. Eine unvollständige Menge an Features im CVS: 3
3. In Bearbeitung: 5
4. Nicht implementiert: 15

Nicht-Compliance. Nach der TrouSerS-Projektseite erfüllt die TSS-Implementierung die TSS-Spezifikation in genau vier Punkten nicht:

1. Policies werden nach Voreinstellung mit einem auf `TSS_SECRET_MODE_NONE` gesetzten Sicherheitsmodus erstellt, anstelle von `TSS_SECRET_MODE_POPUP`. Das bedeutet, wenn Sie in einer GUI Umgebung sind und Ihre Anwendung setzt kein Policy-secret, erhält man ein `TSS_E_POLICY_NO_SECRET`, anstelle eines Popup-Fensters, das nach einem Geheimnis fragt. In einer nichtgraphischen Umgebung in der Ihre Anwendung kein Policy-secret setzt, bekommen Sie `TSS_E_POLICY_NO_SECRET`, anstatt `TSS_E_INTERNAL_ERROR`, wenn die TSS versucht, ein Popup zu starten. Die Anwendung kann ihre Policy natürlich selbst auf `TSS_SECRET_MODE_POPUP` setzen.
2. In `Tspi_TPM_SetStatus` kann man nach Voreinstellung das physische Anwesenheits-Kommando an den TCS leiten. Der TCS prüft, in welchem Runlevel sich die Maschine befindet und wenn es sich um den Single-User-Modus handelt, darf das physische Anwesenheitskommando ausgeführt werden. Das befähigt die Hardware ohne BIOS-Unterstützung des TPMs, ihn zu Aktivieren, Deaktivieren oder zu Löschen.
3. Der 1.1 TrouSerS TSS (Version 0.2.5+) unterstützt Callbacks nach TSS 1.2. In der TSS 1.1b Spezifikation werden Callbacks unter der Verwendung von einer konstanten Breite gesetzt, ein 32-bit Feld in `Tspi_SetAttribUint32`. Dies hält 64-bit Anwendungen davon ab, Callbacks zu setzen. Um eine 64-bit TSS 1.1 Anwendung zu aktivieren bietet TrouSerS Unterstützung für Callbacks nach TSS 1.2.
4. Der 1.1 TrouSerS TSS (Version 0.2.5+) unterstützt den TSS 1.2 Attribute zur Kontrolle der NULL-Terminierung der Daten von Passwörtern.

Obwohl TrouSerS die Spezifikation wie oben beschrieben nicht erfüllt, ist es möglich, eine strikte Spezifikationserfüllung zu aktivieren, indem der Quelltext wie folgt übersetzt wird:

```
./configure --enable-strict-spec-compliance
```

3.1.3. OpenSSL-TPM-Engine

Die OpenSSL-TPM-Engine enthält die folgenden kryptographischen Funktionen:

- RSA Public Key Verschlüsselung, RSA Private Key Entschlüsselung, RSA Signatur, RSA Überprüfung und eine RSA Schlüsselerzeugung unter Verwendung des TPMs
- Zufallszahlenerzeugung unter Verwendung des TPMs
- Laden öffentlicher und privater RSA Schlüssel
- Drei Kommandos zur Konfiguration

Die drei unterstützten Konfigurationskommandos sind (i) `TPM_CMD_SO_PATH` um den Pfad zu der verwendeten Bibliothek `libtspi.so` zu definieren, (ii) `TPM_CMD_PIN` um das

SRK-Geheimnis zu setzen und (iii) `TPM_CMD_SECRET_MODE` um den TSS secret-mode, der für alle Geheimnisse verwendet wird, auszuwählen.
Aufgrund der Beschränkung der kryptographischen Hardware innerhalb des TPM können andere asymmetrische und symmetrische Verschlüsselungssysteme nicht durch das TPM unterstützt werden. Trotzdem wird weder die Verwendung der TPM-eigenen SHA-1-Engine (mittels `ENGINE_DIGESTS_PTR`) noch der TPM-eigene nichtflüchtige Speicher (mittels `STORE_METHOD`) unterstützt.

3.1.4. TPM-Manager

Die aktuelle stabile Version des TPM-Manager ist Version 0.4. Sie basiert auf TrouSerS Version 0.2.x, welche die das TSS-Interface Version 1.1b realisiert. Obwohl es möglich ist Version 0.4 des TPM-Managers mit TrouSerS Version 3.x zu kompilieren und zu verwenden, ist es nicht empfehlenswert, solange viele Funktionen aufgrund des veränderten Verhaltens der TrouSerS-Implementierung nicht funktionieren. Um TPMs der Version 1.2 zu testen, wurde die letzte Entwicklerversion des TPM-Managers 0.5 und TrouSerS 0.3.1 verwendet.
Die vom TPM-Manager zur Verfügung gestellten Funktionen können in folgende Funktionspakete aufgeteilt werden:

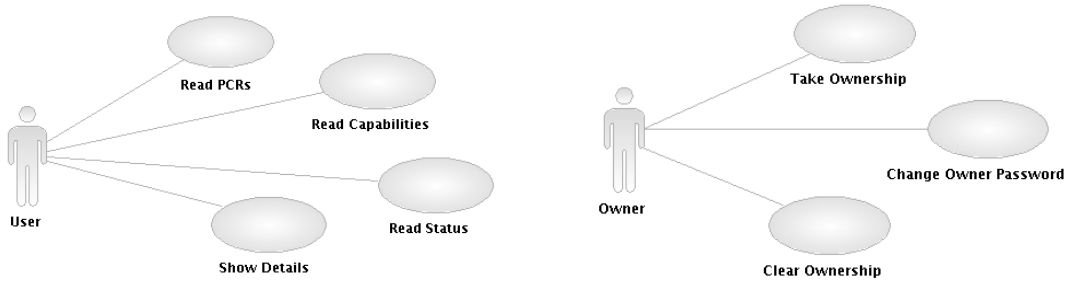
- TPM Info: Generische Information über den aktuellen Zustand des TPM.
- Owner Settings: Benutzerbezogene TPM-Operationen.
- TPM Settings: TPM-spezifische Operationen.
- Maintenance: Backup und Wiederherstellung des TPM Zustands.
- EK Management: Endorsement-Key Management Operationen.

Im Folgenden werden die TPM-Management-Funktionen, die die o.g. Funktionspakete bereitstellen, genauer erklärt.

TPM Info. Dieses Funktionspaket liefert grundlegende Informationen über den TPM-Status, d. h. welcher TSS und TPM-Treiber gefunden wurde, Existenz und Typ des Endorsement-Schlüssels, TPM-Ressourcen und den gegenwärtigen PCR-Status (vgl. Abbildung 3.1). Der Status des TPM-Treibers ist aktuell nicht veränderbar, solange die TSS-Spezifikation keine Informationen über den Zugang des TPM-Treibers liefert. Deshalb nimmt der TPM-Manager an, dass ein echter TPM-Treiber existiert, falls der TCS-Dämon gefunden wird. Darüberhinaus zeigt Version 0.4 des TPM-Managers nur die Zahl der verfügbaren PCRs als Ressourcen.

Owner settings. Hier kann der Benutzer das TPM übernehmen („Take Ownership“), das Owner-Passwort ändern und einen bestehenden Besitzer (owner) löschen (vgl. Abbildung 3.1). Um Kompatibilität zu parallelen Windows-Installationen zu ermöglichen, bietet die Developer-Version des TPM-Managers die Möglichkeit, das

`TSS_WELL_KNOWN_SECRET` als SRK-Passwort bei der Übernahme zu verwenden.

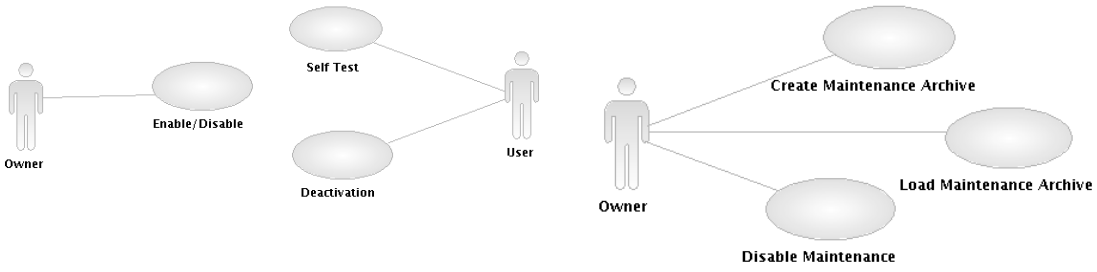


(a) TPM Info (b) TPM Owner

Abbildung 3.1.: Anwendungsfälle realisiert durch die Funktionspakete TPM Info (a) und TPM Owner (b).

TPM settings. Dieses Funktionspaket erlaubt es dem Benutzer, einige TPM-spezifische Einstellungen vorzunehmen, etwa, das TPM zu aktivieren, deaktivieren und auszuschalten, die TPM-internen Testfunktionen aufzurufen und die TPM-Firmware zu aktualisieren (vgl. Abbildung 3.2). Obwohl einige TPM-Modelle (etwa einige von Infineon) Updates von Firmware erlauben, ist diese Funktion in der stabilen Version des TPM-Managers noch nicht vollständig implementiert.

Maintenance. TPM-Maintenance beinhaltet die Möglichkeit, TPM-Archive zu erstellen (d. h. Backups vom SRK), ein Archiv zu Laden und diese Funktionalität dauerhaft zu deaktivieren (vgl. Abbildung 3.2). Jedoch ist TPM-Maintenance ein optionales Feature der TPM-Spezifikation und nach vorliegenden Kenntnissen nicht implementiert. Daher kann derzeit diese Funktionalität auch nicht benutzt oder getestet werden.



(a) TPM Settings (b) TPM Maintenance

Abbildung 3.2.: Der Anwendungsfall, der durch die Funktionspakete TPM-Settings (a) und Maintenance (b) realisiert ist.

EK Management. Dieses Funktionspaket liefert die Funktionen zur Handhabung des Endorsement Key. Bereitgestellte Funktionen sind Erstellung, Löschen und Lesen des EK, Lesen des EK-Zertifikats sowie die Möglichkeit, die Lesefunktion ohne Besitzerautorisierung zu deaktivieren (vgl. Abbildung 3.3). Erstellen und Löschen des Endorsement Key ist zur Zeit nicht implementiert.

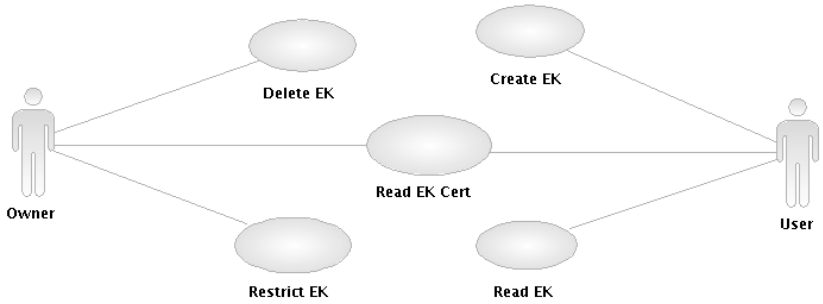


Abbildung 3.3.: Realisierte Anwendungsfälle durch das Funktionspaket EK-Management.

3.2. Testumgebung und Testfälle

Im Folgenden werden die Testumgebungen und die Testfälle, die zur Evaluierung der Interoperabilität der Sicherheitsarchitekturen verwendet werden, beschrieben. Die Tests basieren auf den in Kapitel 2 analysierten OSS-TC-Komponenten.

3.2.1. Testumgebung

Alle Interoperabilitätstests wurden auf einem Notebook Modell HP Compaq 6715b mit Infineon TPM Version 1.2 mit 2GB RAM und dem neusten offiziellen TrouSerS Release (Version 0.3.1) durchgeführt. Darüberhinaus liefern die drei im Folgenden beschriebenen Betriebssystemarchitekturen die Basis für den Test der Interoperabilität.

3.2.1.1. Security-Enhanced Linux (SE-Linux)

Security-Enhanced Linux (SE-Linux) ist eine Erweiterung von Linux, die eine Auswahl an Sicherheitsrichtlinien durch die Verwendung von Linux-Sicherheits-Modulen (LSM) erzwingen kann [19, 18]. SE-Linux wurde maßgeblich von der National Security Agency (NSA) (und anderen wie RedHat, Department of Defense (DoD), IBM, HP oder Tresys) entwickelt und beinhaltet ein Kernmodul, Patches zu verschiedenen sicherheitsrelevanten Anwendungen, sowie Sicherheitsrichtlinien (sog. *security policies*).

In einem herkömmlichen diskreten Linux Sicherheitsmodell (also welches Discretionary Access Control (DAC) unterstützt) kann jedes Programm seine eigenen Ressourcen

kontrollieren. Im Gegensatz dazu erlaubt SE-Linux eine genauere Zugangskontrolle, um systemweit Sicherheitsregeln durchzusetzen (Mandatory Access Control (MAC)). Realisiert wird dies durch das Prinzip der minimalen Berechtigungen (*least privileges*), d.h. einem SE-Linux Prozess können genau die Rechte zugewiesen werden, die gerade zum Funktionieren benötigt werden.

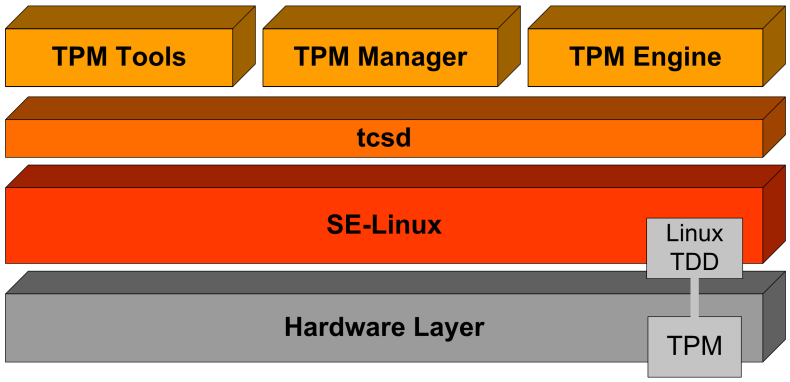


Abbildung 3.4.: SE-Linux basierte Architektur.

SE-Linux basiert auf der FLASK-Sicherheitsarchitektur [20], die ursprünglich dazu entwickelt wurde, um einige MAC-eigenen Probleme der Policy-Durchsetzung zu überarbeiten. FLASK bietet generelle Unterstützung zur Durchsetzung vieler Arten von MAC-Policies, einschließlich jener, welche auf den Konzepten Type-Enforcement, rollenbasierter Zugangskontrolle und Multilevelsicherheit basieren.

Das erste Release von SE-Linux ist im Dezember 2000 erschienen und basierte auf Linux Version 2.2.12 und den Red Hat Version 6.1 Utilities. Heute ist SE-Linux in vielen Linux-Distributionen integriert und kann unter http://www.nsa.gov/selinux/ heruntergeladen werden. SE-Linux beinhaltet für jeden Zweck Security-Policy-Konfigurationen, die dazu entwickelt wurden, um verschiedene Sicherheitsziele zu erreichen. Die Flexibilität des Systems erlaubt es, die Policy zu verändern oder zu erweitern, um eine bestimmte geforderte Policy einer gegebenen Installation individuell anzupassen.

Die auf SE-Linux basierte TC-Architektur wird in Abbildung 3.4 dargestellt. Die Testumgebung basiert auf einer um SE-Linux erweiterten Gentoo[21] 2007.0-hardened Installation.
Zusätzlich wurde Version 0.3.1 des TrouSerS Repository installiert. Die vollständige Architektur wurde unter Verwendung von TrustedGRUB Version 1.1.3 gebootet. Weiterhin wurde das SE-Linux-System unter Verwendung der Sicherheitspolicy `targeted` getestet, da es sich hierbei um einen für Desktopsysteme optimierten Satz von Zugriffsregeln handelt.
Anhang C beschreibt allgemein, wie ein reguläres Gentoo System in ein gehärtetes SE-Linux-System umgewandelt werden kann.

21 Siehe http://www.gentoo.org/proj/en/hardened/selinux/index.xml

3.2.1.2. Der Xen-Hypervisor

Xen ist ein freier Virtual Machine Monitor (VMM), auch bekannt als Hypervisor [3, 12]. Im Gegensatz zu anderen VMMs wie VMware wird dieser monolithische Hypervisor direkt auf der Hardwareschicht ausgeführt. Xen besitzt keine eigenen Hardwaretreiber. Stattdessen benutzt Xen Gerätetreiber einer privilegierten Linux-Instanz *Dom-0*. *Dom-0* enthält auch die Management-Tools für den Hypervisor, genannt Xen-Tools.

Neben *Dom-0* können weitere virtuelle Maschinen (*guests*), etwa *Dom-U*, parallel ausgeführt werden. Die Xen-basierte Architektur ist in Abbildung 3.5 gezeigt.

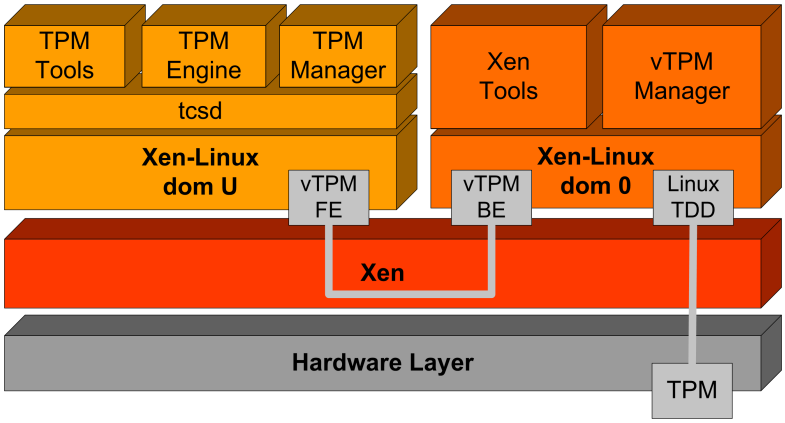


Abbildung 3.5.: Xen-basierte Architektur.

Seit Version 3.0 bietet Xen eine erste Implementierung einer TPM-Virtualisierung [4]. Virtuelle TPM-Geräte werden unter Verwendung passender Frontends (vTPM FE) und einem Backend (vTPM BE) zu jeder Gastdomäne exportiert. Die Kommunikation zwischen ihnen wird auf Basis des Xen-Bus realisiert. Die vTPM-Frontends durch den vTPM-Manager, der als `vtpm_managerd`-Dämon implementiert ist und in *Dom-0* ausgeführt wird, verwaltet. Die Zustände der virtuellen TPMs der Gastdomänen werden mittels '*sealing*' an das echte Hardware-TPM gebunden, das durch gewöhnliche Gerätetreiber innerhalb des *Dom-0*-Linux zugänglich ist.

Sowohl *Dom-0*-Linux als auch jede Gastdomäne (wie *Dom-U*) verwenden eine eigene TSS-Instanz um auf das TPM bzw. vTPM zuzugreifen.

3.2.1.3. Der Turaya-Sicherheitskern

Der Sicherheitskern Turaya bietet seinen Anwendungen sowohl starke Isolation als auch mehrseitige Durchsetzung von Sicherheitsrichtlinien [16, 1]. Wie in Abbildung 3.6 gezeigt, besteht eine Turaya-basierte Softwarearchitektur aus den folgenden drei Schichten: (i) einer Hardwareschicht, die konventionelle Hardware wie Speicher, CPU, Geräte und das TPM

enthält; (ii) der Turaya-Sicherheitskern bestehend aus einer Hypervisor-Schicht und einer Security-Service Schicht; (iii) Anwendungen und Betriebssysteme, die parallel und isoliert voneinander ausgeführt werden.

Hypervisor-Schicht. Die Hypervisor-Schicht des Sicherheitskerns arbeitet wie ein traditioneller VMM, indem er Hardwareressourcen verwaltet und die Unterstützung grundlegender Virtualisierung bereitstellt. Entsprechend des modularen Konzepts der Turaya-Architektur können verschiedene Hypervisor und Mikrokerne als Hypervisor-Schicht verwendet werden. In diesem Testszenario wurde ein Turaya-Sicherheitskern, der auf dem L4-Mikrokern (vgl. Abbildung 3.7) basiert, verwendet. Ein Mikrokern ist ein minimales Betriebssystem, welches keine eigenen Betriebssystemdienste bereitstellt. Stattdessen wird lediglich der Mechanismus zur Verfügung gestellt, mit dem so einen Dienst implementiert werden kann, wie etwa Low-Level-Addressraummanagement, Thread-Management und Inter-Process Communication (IPC). Auf dem Mikrokern werden elementare Ressourcen-Management-Dienste ausgeführt, wie z. B. Prozess-Management, Speicher-Management oder Interrupt-Management. Die Dienste der Hypervisor-Schicht stellen auch Gerätetreiber zur Verfügung und setzen die systemweiten Sicherheitsrichtlinien durch.

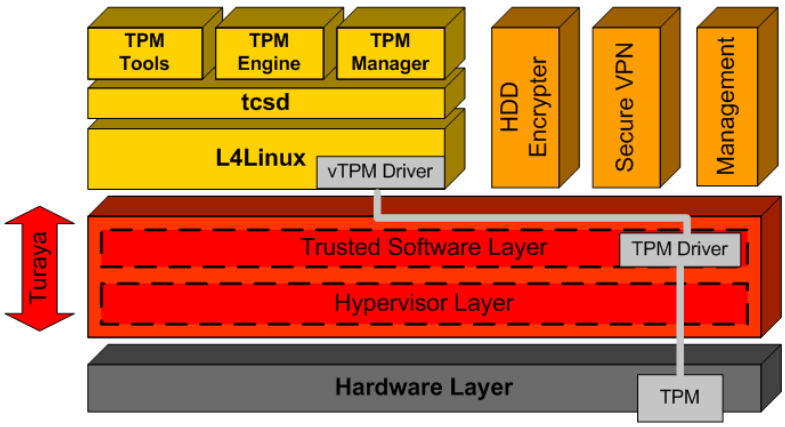


Abbildung 3.6.: Turaya-basierte „Trusted Computing"-Architektur.

Trusted-Software-Schicht. Über der Hypervisor-Schicht bietet die Trusted-Software-Schicht abstraktere Sicherheitsdienste. Beispielsweise stellt die sichere Anwenderschnittstelle (Trusted GUI) einen vertrauenswürdigen Pfad zwischen Anwendung und Anwender zur Verfügung. Der Compartment-Manager verwaltet die Erstellung, Aktualisierung und Löschung von Compartments. Darüberhinaus kontrolliert er, welche Compartments gestartet werden dürfen, und er erzwingt die Durchsetzung der Sicherheitsrichtlinien. Der Speichermanager stellt für andere Compartments persistenten Speicher bereit und schützt die Integrität, Vertraulichkeit, Verfügbarkeit und Aktualität der gespeicherten Daten. Darüberhinaus erzwingt der Speichermanager eine strikte Isolierung

zwischen Compartments, indem er die gespeicherten Daten an die Konfiguration der jeweiligen Compartments bindet.
Die Turaya-basierte Testumgebung beinhaltet eine Linux-Instanz, die oberhalb der Trusted-Software-Schicht ausgeführt wird. Diese Linux-Instanz verwendet einen TPM-Treiber-„Stub“ (vTPM driver) um den TPM-Multiplexer (TPM driver) zu benutzen, der Teil der Trusted-Software-Schicht ist. Der Linux vTPM-Treiber kommuniziert mit dem Turaya-TPM-Treiber unter Verwendung des IPC-Mechanismus, den der darunterliegende Mikrokern bereitstellt. Auf dem Linux vTPM-Treiber werden TrouSerS, die TPM-Tools und der TPM-Manager ausgeführt. Die ganze Architektur wird unter Verwendung des TrustedGRUB Version 1.1.3 gebootet.

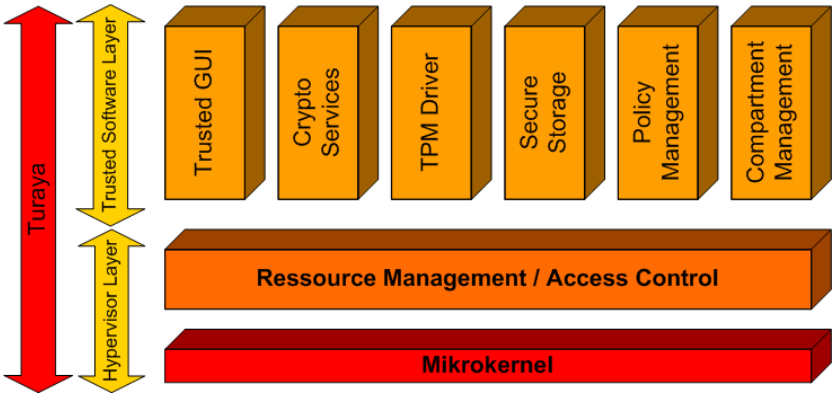


Abbildung 3.7.: Komponenten der Turaya-Sicherheitsschicht.

3.2.2. Testfälle

Um die Interoperabilität der Implementierungen zu prüfen, werden die folgenden Funktionstest basierend auf den oben beschriebenen Softwarearchitekturen durchgeführt:

- Auf dem TPM-Manager basierende Tests:
  - TC.1 Übernahme („Take Ownership“)
  - TC.2 Änderung des Besitzerpasswortes
  - TC.3 Deaktivierung des TPM
  - TC.4 Abschalten des TPM
  - TC.5 Aktivieren des TPM

- Tests basierend auf den TPM-Tools von TrouSerS:
  - TC.6 Versiegeln (*sealing*) einer Datei
  - TC.7 Dateiversiegelung in passender Konfiguration rückgängig machen (*unsealing*)

- Auf TrouSerS OpenSSL-TPM-Engine basierende Tests
  - TC.8 Erstellung eines TPM-geschützten Schlüssels

- TC.9 Signierung einer Datei unter Verwendung des TPM-geschützten Signaturschlüssels

Anmerkung. Die TPM-Tools beinhalten einen Befehl `tpm_seal` zur Versiegelung von Dateien, aber kein passendes Werkzeug, um die Versiegelung wieder rückgängig zu machen. Um obige Tests ausführen und die Korrektheit der Versiegelungsfunktion testen zu können, wurde der Befehl `tpm_unseal` implementiert. Der Quelltext dieses Befehls steht in Anhang A.

3.3. Interoperabilitätstests

Im Folgenden werden die Ergebnisse der durchgeführten Tests zur Interoperabilität zusammengefasst. Zudem werden weitere Stolperstellen und passende Lösungen diskutiert.

3.3.1. SE-Linux-basierte Architektur

Wie schon in Abschnitt 3.2.1.1 beschrieben, erzwingt eine SE-Linux-basierte Rechnerplattform das Least-Privilege-Prinzip, indem ein Dämon oder eine Anwendung nur die Rechte gewährt bekommt, die wirklich benötigt werden.
Mit dem installierten Gentoo SE-Linux-System in dem wie in Abschnitt 3.2.1.1 und Anhang C beschriebenen Enforcement-Modus '`targeted`' konnten alle oben beschriebenen Tests erfolgreich durchgeführt werden. Um die auf dem TPM-Manager basierenden Tests durchzuführen, musste ein graphisches Benutzersystem (XServer + KDE) und der TSS Core Service-Dämon gestartet werden. Das graphische System konnte nur nach der Installation der SE-Linux-Policy `sec-policy/selinux-desktop` starten. Der `tcsd` startete ohne Änderungen erfolgreich.

Wenn SE-Linux im Enforcement-Modus '`strict`' ausgeführt wird, ist es möglich den `tcsd` nach der Installation von zusätzlichen Zugriffsregeln zu starten. Die notwendigen Regeln kommen von der TrouSerS-Distribution (geschrieben für Fedora Core 5) und werden in Listing 3.2 und Listing 3.4 aufgezeigt. Die Installationsroutine für Fedora Core wird in Tabelle 3.3 beschrieben.
In Gentoo Linux haben sich die Routinen zur Installation von Zugriffsregeln seit 2006 geändert. Regeln werden nun nicht mehr durch ein `Makefile` gesetzt, sondern als ladbare Module kompiliert. Dazu existieren neue Werkzeuge unter Gentoo, `checkmodule`, `semodule`, `semodule_package` und `audit2allow`. Eine einfachere, aber nicht sehr sichere Möglichkeit besteht darin, die notwendigen Regeln automatisch durch `audit2allow` generieren zu lassen.
Um das zu tun, muss SE-Linux neu gebootet und TrouSerS neu gestartet werden. Der Startvorgang wird fehlschlagen, aber es ist möglich die erforderlichen Rechte zu prüfen und ein neues Regelwerk für den `tcsd` wie folgt zu erstellen:

```
// Creating ruleset:
# audit2allow -d -M tcsd
// Installing new ruleset:
# semodule -i tcsd.pp
```

Danach arbeiten TrouSerS' `tcsd` und die TPM-Tools ohne Probleme. Allerdings muss darauf geachtet werden, dass dies ein sehr generischer Weg ist gültige Regelsätze für Gentoo zu erstellen. `audit2allow` überprüft lediglich welche Zugriffe verweigert wurden und gewährt die erforderlichen Rechte automatisch. Aber diese Rechte könnten zu allgemein sein, daher ist es sinnvoller, das spezifischere Fedora `Makefile`-Regelwerk an das modulare System von Gentoo anzupassen.

Obwohl es möglich ist einen TPM-Gerätetreiber zu Laden, TrouSerS zu starten und die TPM-Tools im '`strict`'-Modus von SE-Linux zu verwenden, ist es nicht möglich, den TPM-Manager zu verwenden, solange es nicht möglich ist, sich am XServer einzuloggen. Während der Überprüfung des eingegeben Benutzerpasswortes kommt es zu einer Sicherheitsverletzung beim Zugriff auf die Datei `/etc/shadow`, daher kann sich der Benutzer ohne weitere Systemmodifikationen nicht am XServer anmelden[22] .

Anmerkung: Um die Sicherheitsrichtlinien des Systems zu setzen, wird das gesamte Dateisystem etikettiert (*labeling*) und so für jede Datei einzelne Richtlinien und Kontexte vergeben. Jedoch kommt es unter Gentoo bei der Verwendung von `udev`, welches die Device-Nodes in `/dev/` dynamisch erzeugt, zu Startproblemen, da die neu erzeugten Dateien noch nicht etikettiert sind. Dies führt dazu, dass SE-Linux nicht die korrekten Rechte beim Starten im '`strict`'-Modus hat und sich daher ein Benutzer nicht am System anmelden kann, da er keine Rechte auf die Dateien `console` und `getty` besitzt. Dieses Problem kann dadurch gelöst werden, indem Gentoo auf statische Device-Nodes umkonfiguriert wird. Dazu wird die Konfiguration `RC_DEVICES` in `/etc/conf.d/rc` auf `RC_DEVICES=static` gesetzt. Nach einem Neustart des Systems, muss das Dateisystem erneut etikettiert werden mit den Befehl

```
rlpkg –a –r.
```

Da nun `udev` nicht mehr läuft, müssen das TPM-Device-Node manuell mit

```
mknod /dev/tpm0 c 10 224
```

erstellt und die Rechte mit

22 Es ist fraglich, ob es überhaupt sinnvoll ist, einen XServer in einer streng SE-Linux-geschützten Umgebung zu erlauben.

```
chown tss:tss /dev/tpm0
```

gesetzt werden. Zusätzlich ist unter Gentoo die globale boolsche Variable `global_ssp` auf '*false*' voreingestellt, obwohl diese '*true*' sein muss. Der Variablenwert kann mit dem folgenden Kommando gesetzt werden:

```
setsebool -P global_ssp 1
```

Listing 3.2: trousers.fc Security Policy für Fedora

```
/usr/sbin/tcsd system_u:object_r:tcsd_exec_t
/etc/tcsd.conf system_u:object_r:tcsd_config_t
/var/lib/tpm(/.*)? system_u:object_r:tcsd_readwrite_t
/dev/tpm(.*) system_u:object_r:tcsd_device_t
```

Die folgende Ausgabe zeigt das Ergebnis nach erfolgreichem Laden der Sicherheitspolicies:

```
# ls -Z /dev/tpm*
crw-rw---- root root system_u:object_r:tcsd_device_t  /dev/tpm0
# ps -Zef | grep tcsd
root:system_r:tcsd_t root 16362 1 0 15:10 ?00:00:00 /usr/sbin/tcsd
```

Listing 3.3: Installation der TrouSerS SE-Linux Sicherheitspolicies in Fedora

```
# cp ./dist/fedora/trousers.te \
  /etc/selinux/targeted/src/policy/domains/program
# cp ./dist/fedora/trousers.fc \
  /etc/selinux/targeted/src/policy/file_contexts/program
# cd / etc / selinux / targeted / src / policy
# make clean
# make reload
# make install
# make relabel
```

Listing 3.4: trousers.te Security Policy für Fedora

```
type tcsd_device_t , device_type , dev_fs ;
type tcsd_readwrite_t , file_type ;
type tcsd_config_t , file_type , sysadmfile ;
daemon_domain ( tcsd , ‘ )
general_domain_access ( tcsd_t )
allow unconfined_t tcsd_t : process transition ;
type_transition unconfined_t tcsd_exec_t : process tcsd_t ;
allow tcsd_t tcsd_exec_t : dir r_dir_perms ;
allow tcsd_t etc_t : file { read getattr lock ioctl } ;
allow tcsd_t etc_t : lnk_file { read getattr } ;
allow tcsd_t devtty_t : chr_file { ioctl read getattr lock write append } ;
allow tcsd_t devpts_t : chr_file { ioctl read getattr lock write append } ;
can_network ( tcsd_t )
read_sysctl ( tcsd_t , full )
r_dir_file ( tcsd_t , usr_t )
r_dir_file ( tcsd_t , tcsd_config_t )
rw_dir_file ( tcsd_t , tcsd_readwrite_t )
allow tcsd_t tcsd_readwrite_t : file { setattr } ;
allow tcsd_t tcsd_readwrite_t : dir { setattr } ;
allow tcsd_t tcsd_device_t : chr_file { ioctl read getattr lock write append } ;
allow tcsd_t { random_device_t } : chr_file { read getattr } ;
allow tcsd_t lib_t : dir r_dir_perms ;
allow tcsd_t lib_t : file { rx_file_perms execmod } ;
allow tcsd_t lib_t : lnk_file r_file_perms ;
allow tcsd_t lib_t : file { rx_file_perms execmod } ;
allow tcsd_t lib_t : lnk_file r_file_perms ;
allow tcsd_t lib_t : file { rx_file_perms execmod } ;
allow tcsd_t lib_t : lnk_file r_file_perms ;
allow tcsd_t var_lib_t : dir r_dir_perms ;
allow tcsd_t var_lib_t : file { rx_file_perms execmod } ;
allow tcsd_t var_lib_t : lnk_file r_file_perms ;
allow tcsd_t port_type : tcp_socket { send_msg recv_msg name_bind } ;
allow tcsd_t self : capability { chown net_bind_service dac_override fowner
fsetid
```

3.3.2. Xen-basierte Architektur

Die derzeitige Implementierung des vTPM-Manager realisiert die Schnittstellen und Befehle der TPM-Spezifikation 1.1b. Aufgrund einiger gelöschter Befehle in der v1.2 TPM-Spezifikation ist die Verwendung von TPMs der Version 1.2 derzeit nicht möglich. Deshalb ist es auch nicht möglich, das TPM zu virtualisieren. Somit fehlen die fundamentalen Grundlagen, um die Interoperabilitätstests aus Abschnitt 3.2.2 durchzuführen.

Um dennoch Tests auf Basis einer Xen-Architektur durchzuführen, wurden die Interoperabilitätstests auf einem IBM Thinkpad T40 mit einem Atmel TPM nach Spezifikation v1.1b durchgeführt. Auf diese Plattform kam Xen v3.1.0 zum Einsatz. Weiterhin wurde TrouSerS 0.2.9 in *dom-U* installiert, da TPMs der Version 1.2 durch den

vTPM-Manager von Xen nicht unterstützt werden. Die komplette Architektur wurde unter Verwendung von TrustedGRUB Version 1.1.3 gebootet.

Der folgende Absatz fasst die Ergebnisse der durchgeführten Interoperabilitätstests einschließlich der Aufgedeckten Fallstricke und geeigneter Lösungen zusammen.

- **vTPM und vTPM Manager** *(Dom-0)*:
  Der vTPM-Manager von Xen konnte erfolgreich gestartet werden. Wenn eine Gastdomäne von Xen gebootet wird, wird ein neues vTPM erstellt und mit dem Gastbetriebssystem verbunden. Da dieses vTPM nach jedem Start der Gastdomäne neu erstellt wird, muss nach jedem Neustart der Gastmaschine `tpm_takeownership` ausgeführt werden, um das vTPM in einen '`owned`' Status zu bringen. Wenn der vTPM-Manager korrekt gestartet und der zugehörige Linux vTPM-Gerätetreiber geladen ist, verhält sich das vTPM wie ein normales 1.1b TPM.
- **TrouSerS** *(Dom-U)*:
  Der `tcsd` von TrouSerS in der Gastdomäne startet wie erwartet.
- **TPM Tools** *(Dom-U)*:
  Aufgrund einiger Fehler in früheren Versionen vom TrouSerS TSS Version 0.2.9, der für die Tests unter Xen verwendet werden musste, konnte `tpm_sealdata` nicht korrekt ausgeführt werden, da die Authentisierungsdaten nicht korrekt übergeben wurden. Zusätzlich werden Ressourcen (beispielsweise geöffnete Sitzungen), innerhalb der TPM-Hardware nicht korrekt beendet. Das führt zu nicht ausreichenden Ressourcen innerhalb des TPM nach nur wenigen Befehlen. Die Ressourcen mussten entweder explizit oder implizit durch einen Neustart der (Gesamt-)Maschine freigegeben werden.
- **TPM Manager** *(Dom-U)*:
  Der TPM-Manager, der innerhalb einer Xen-virtualisierten Gastdomäne läuft, konnte nicht den wahren Status des vTPM herausfinden und hat daher angenommen, dass das vTPM abgeschaltet ist.
- **OpenSSL TPM Engine** (*Dom-U*):
  Alle Tests verliefen erfolgreich.

3.3.3. Turaya-basierte Architektur

Im Gegensatz zum vTPM von Xen bietet das TPM-Framework des Turaya-Sicherheitskern kein virtuelles TPM für parallel ausgeführte Betriebssysteminstanzen an. Stattdessen bildet Turaya die vom TPM bereitgestellte Funktionalität auf verschiedene Sicherheitsdienste ab, die einen höheren Abstraktionslevel der bereitgestellten TC-Funktionalität liefern. Beispielsweise werden monotonischen Zähler, die ein TPM seit Version 1.2 bietet, auf den sicheren Speicherdienst abgebildet, der aktuellen (*freshness*) und persistenten Speicher zur Verfügung stellt. Obwohl dieses Design die Realisierung von sicherheitskritischen

Anwendungen sehr effektiv erlaubt, ist die gleichzeitige Verwendung eines virtuellen TPMs durch verschiedene Betriebssysteme derzeit nicht möglich. Jedoch ist nach dem Turaya Meilensteinplan eine virtuelle TPM-Implementierung basierend auf den bereitgestellten Sicherheitsdiensten derzeit in der Entwicklung.

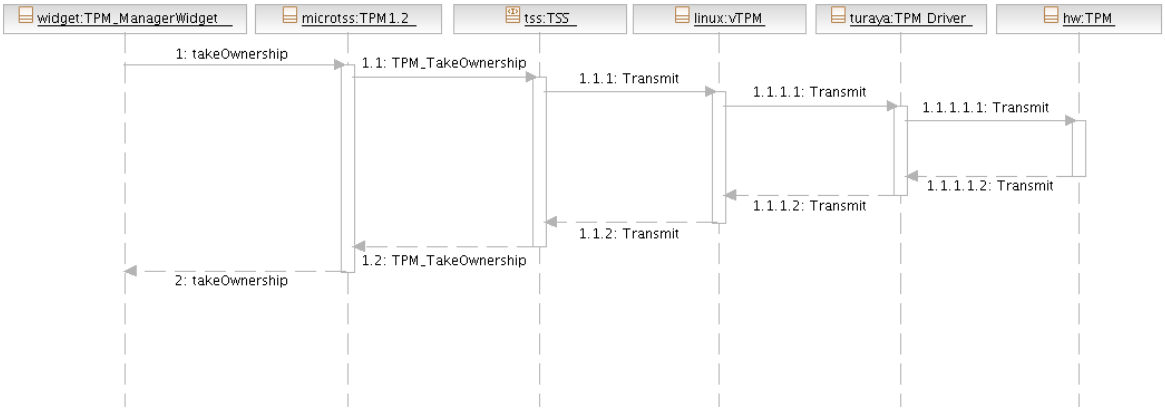


Abbildung 3.8.: Datenfluss nach dem Aufruf von „Take Ownership“ von einem Turaya-basierten System.

Abbildung 3.8 zeigt eine stark vereinfachte Skizze des Nachrichtenflusses, nachdem eine „Take Ownerschip“-Operation des TPM-Manager über dem Turaya-Sicherheitskern aufgerufen wurde. Der Linux TPM-Treiber leitet den TPM-Befehl per IPC weiter an den Turaya-TPM-Treiber, welcher den Befehl selbst an das physikalische TPM leitet.

Da der TPM-Treiber von Turaya - hier implementiert als isolierter Dienst innerhalb der Turaya-Sicherheitsschicht (vgl. Abbildung 3.7) - TPM-Befehle direkt an das physikalische TPM leitet, waren keine Probleme in der Interoperabilität zu erwarten. In der Tat verliefen die Prüfungen der Interoperabilität basierend auf den TPM-Tools, dem TPM-Manager und der OpenSSL-TPM-Engine erfolgreich.

4. Zusammenfassung

Diese Studie bietet einen Überblick über bestehende OSS Software, die „Trusted Computing“-Technologie - wie von der Trusted Computing Group spezifiziert - unterstützt, evaluiert ihre Kompatibilität und identifiziert Mängel und fehlende Komponenten, die zur Realisierung einer „Trusted Computing“-Plattform erforderlich sind.

Die getesteten und analysierten Softwarekomponenten sind verschiedene Bootloader mit TCG-Unterstützung, der Linux-TPM Device Driver, der TrouSerS TCG Software Stack, die TrouSerS TPM-Tools, die OpenSSL-TPM-Engine und der TPM-Manager.

Weiterhin wurde die Interoperabilität und Funktionalität der hier vorgestellten Software auf den folgenden Sicherheits-Architekturen evaluiert: Security-Enhanced Linux, Xen-Hypervisor und dem Turaya-Sicherheitskern.

4.1. „Trusted Computing“-Bausteine

Nur einige wenige Funktionen der evaluierten OSS-Lösungen sind entweder nicht vorhanden oder noch nicht komplett funktionsfähig. Die Analysen dieser Studie zeigen, dass die meisten Bausteine, sprich ein TCG-kompatibler Bootloader, ein TPM-Treiber-Framework und eine Implementierung der TCG Software Stack verfügbar und stabil genug zur Verwendung in sicherheitskritischen Diensten und Anwendungen sind.

Es ist jedoch offenkundig, dass bis zum heutigen Zeitpunkt keine Anwendung existiert, die von dieser Technologie Gebrauch macht. Sogar die einfachsten Anwendungen, etwa die Verwendung eines TPM zur Erzeugung von Zufallszahlen innerhalb eines GNU/Linux Betriebssystems wird bisher noch nicht realisiert. Theoretisch sollte das OpenSSL-Engine-Konzept zusammen mit der OpenSSL-TPM-Engine einen leichten Weg bieten, existierende Anwendungen derart zu migrieren, dass sie die vom TPM angebotenen Funktionalitäten nutzen können, da das OpenSSL-Engine-Konzept die Erweiterung um zusätzliche kryptographische Module erlaubt. Unsere Erfahrungen zeigen jedoch, dass Anwendungen auf das Engine-Konzept von OpenSSL in der Praxis nicht so leicht zu migrieren sind, da viele Änderungen am Sourcecode nötig sind.

4.2. „Trusted Computing“-Architekturen

Die Analyse von „Trusted Computing“-Architekturen, die auf SE-Linux, Xen und Turaya basieren, hat gezeigt, dass die Interoperabilität der existierenden „Trusted Computing“-Lösungen hoch genug ist, um auf ihnen TC-basierte Anwendungen zu realisieren. In diesem Zusammenhang sind die während dieser Analyse identifizierten offenen Punkte wie folgt:

- **Xen**
  Durch die Verwendung des Entwicklungszweigs des TrouSerS TSS-Stack war es möglich, alle ausgewählten Anwendungsfälle, die eine TPM Version 1.2 verwenden, erfolgreich zu testen. Die einzige Ausnahme zum jetzigen Zeitpunkt ist die vTPM-Implementierung des Xen-Hypervisor, die zur Zeit nicht stabil genug ist. Während unserer Tests stürzte das vTPM sehr oft ab und verlor seinen internen Zustand immer, wenn die virtuelle Gastmaschine neu gestartet wurde.
- **Turaya**
  Auch mit dem Turaya-Sicherheitskern war es möglich, alle Anwendungsfälle erfolgreich umzusetzen. Unsere Analyse hat jedoch gezeigt, dass der Turaya-Sicherheitskern bisher keine Unterstützung einer vTPM-Implementierung bietet. Obwohl ein TPM-Treiber existiert, der eine virtuelle Maschine mit dem TPM verbindet, ist es zur Zeit nicht möglich, ein TPM gleichzeitig mit verschiedenen virtuellen Maschinen zu benutzen. Nach Aussagen des Turaya-Entwicklungsteam befindet sich eine virtuelle TPM-Implementierung derzeit in Arbeit.
- **SE-Linux**
  Der auf SE-Linux basierende Interoperabilitätstest zeigte, dass der SE-Linux-Support von verbreiteten Linux-Distributionen gut genug ist, um die ausgewählten Testfälle erfolgreich durchzuführen. Im Gegensatz zu dem '`strict`'-Enforcement-Modus, welcher alle Systemdienste schützt, schützt der bei den Tests verwendete '`targeted`'-Enforcement-Modus allerdings nur bestimmte Systemdienste, wie z. B. den HTTP-Dämon.

Obwohl alle drei „Trusted Computing"-Architekturen zur Realisierung von Anwendungen mit TC-Unterstützung verwendet werden können, ist eine fehlerfreie und stabile TC-Architektur bisher nicht verfügbar. Das EMSCB-Projekt[23] und das OpenTC-Projekt[24] zielen darauf ab, so eine Architektur basierend auf dem Xen-Hypervisor bzw. dem Turaya-Sicherheitskern zu entwickeln. Inzwischen können diese Architekturen schon für spezielle Anwendungen und Szenarien verwendet werden, aber sie sind noch nicht für den universellen Masseneinsatz ausgereift.

Für Linux existiert bisher nur die von IBM entwickelte Integrity Measurement Architecture (IMA) [17]. Sie erlaubt einer entfernten Instanz die Systemkonfiguration zu attestieren, einschließlich aller geladenen Konfigurationsdateien und gemeinsam genutzter Bibliotheken. Allerdings hat die IMA- Architektur auch einige wesentliche Nachteile:

IMA ist eine Erweiterung von Linux, die Measurement Hooks in Funktionen einfügt, die für das Laden von Dateien relevant sind. Immer wenn ein Kernmodul, eine Anwendung oder eine Konfigurationsdatei geladen wird, wird die entsprechende Datei gemessen und der resultierende Hashwert verwendet, um ein PCR zu erweitern. Auf diese Art und Weise setzt IMA die Vertrauenskette vom BIOS und Bootloader bis hinein in die Anwendungs-

23 http://www.emscb.org
24 http://www.opentc.net

ebene fort. Der hauptsächliche Zweck von IMA ist die Analyse der Vertrauenswürdigkeit von geladenen Inhalten basierend auf einer Logdatei, die während des Attestierungsprotokolls bereitgestellt wird. Jedoch erlaubt IMA nicht die Attestierung einzelner Anwendungen. Stattdessen wird die gesamten Rechenplattform (einschließlich Betriebssystem und allen geladenen Anwendungen) attestiert. Darüberhinaus erlaubt IMA nicht das Binden von Daten an bestimmte Anwendungen und Konfigurationen, da sich die PCR-Werte beim Laden einer neuen Datei regelmäßig verändern. Das führt dazu, dass an eine vorherige Konfiguration gebundene Daten nicht mehr zugänglich sind. Weiterhin hängt der Inhalt eines PCRs ebenfalls von der Reihenfolge der geladenen Dateien ab. Wenn sich die Reihenfolge ändert, ändern sich auch die PCR-Werte.

4.3. Offene Fragen

Basierend auf dem Ergebnis dieser Studie sind unserer Meinung nach die folgenden Verbesserungen erforderlich, um eine breitere Verwendung der „Trusted Computing“-Technologie im Open-Source Bereich zu ermöglichen:

4.3.1. Kryptographische API mit TC-Unterstützung

Kurzfristig sollten existierende sicherheitskritische Anwendungen, wie VPN-Clients oder Festplattenverschlüsseler, um TPM-Unterstützung erweitert werden, um von den erweiter ten Funktionalitäten der „Trusted Computing“-Technologie zu profitieren. Allerdings ist die notwendige Erweiterung jeder einzelnen sicherheitsrelevanten Anwendung um „Trusted Computing“ und TSS-Unterstützung nicht realistisch, da die Durchführung signifikante Ressourcen erfordern würde. Das ist umso mehr der Fall, da die TSS-Schnittstelle und die zugehörige TrouSerS Implementierung selbst unter starker Entwicklung stehen.

Unter Microsoft Windows können Anwendungen einheitliche kryptographische Schnittstellen (MS-CAPI) verwenden, die die indirekte Verwendung der TPM-Funktionen erlauben. Abbildung 4.1 beschreibt beispielhaft, wie eine kryptographische API verwendet werden kann, um konventionelle Anwendungen um TSS-Unterstützung zu erweitern. Unter Linux existiert so eine einheitliche kryptographische Schnittstelle bisher nicht. Die momentan einzige Alternative ist die OpenSSL-Engine-Schnittstelle, da OpenSSL ein Quasi-Standard ist, der von vielen sicherheitsrelevanten Anwendungen verwendet wird.

Wie in Abschnitt 2.6 diskutiert, ist die Verwendung der OpenSSL-Engine nicht so einfach und transparent, wie sie sein sollte, da jede Anwendung um die Unterstützung der OpenSSL-Engine erweitert werden muss. Nach unserer Analyse, welche auf verschiedenen OpenSSL-unterstützten Anwendungen basiert (etwa racoon[25] und XSupplicant[26]), würde so eine Erweiterung einen großen Aufwand und eine enge Zusammenarbeit mit den Entwicklern der Anwendungen erfordern.

25 http://ipsec-tools.sourceforge.net/
26 http://open1x.sourceforge.net/

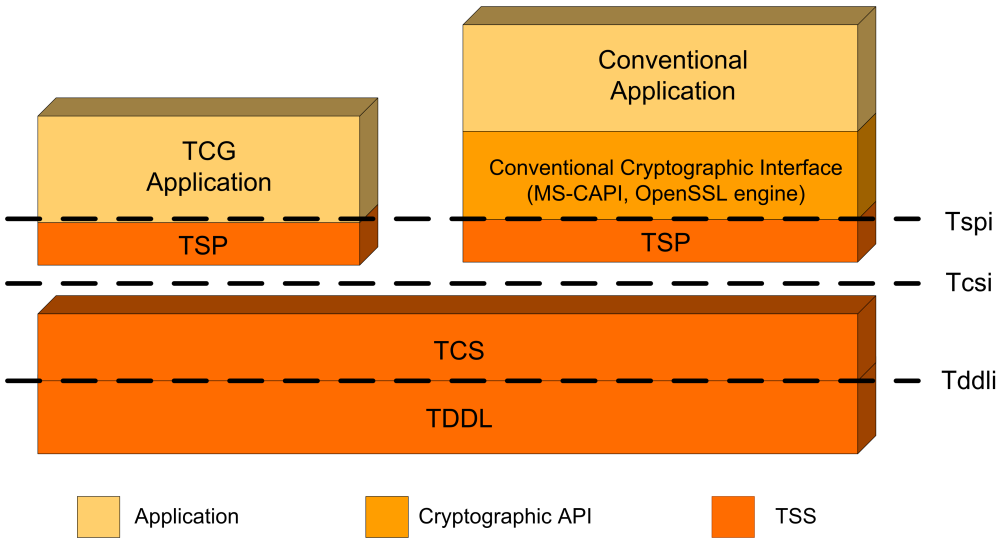


Abbildung 4.1.: Die Verwendung von TSS-Diensten basieren auf einer konventionellen kryptographischen Schnittstelle im Vergleich zu nativen TCG-Anwendungen, welche den TSP direkt verwenden.

Auf lange Sicht wäre es effizienter, das OpenSSL-Engine-Konzept so zu verbessern, dass es transparent arbeitet, ohne die Notwendigkeit jede Anwendung modifizieren zu müssen. Das würde die Verwendung des TPM aus allen Anwendungen heraus erlauben, die OpenSSL unterstützen. Ohne die Möglichkeit der Verwendung von TPM-Funktionen basierend auf einer allgemeinen kryptographischen Schnittstelle wird die TPM-Unterstützung für Linux (und alternative OSS-Lösungen im allgemeinen) nicht vorankommen. Als Ergebnis werden OSS-Lösungen immer weiter proprietären Lösungen hinterherhinken.

4.3.2. Managementlösungen

Ein anderer wichtiger Aspekt TC-basierter Produkte ist die Entwicklung von Managementkonzepten und Werkzeugen, um die im Folgenden beschriebenen Nachteile der TC-Technologie zu kompensieren.

Die Modifikation einer Binärdatei verändert beispielsweise den Hashwert und damit ändern sich auch die PCR-Werte derart, dass gebundene Daten nicht mehr entschlüsselt werden können. Dies kann im Fall einer bösartigen Modifikation erwünscht sein, jedoch wird auch die notwendige Systemaktualisierung und das Einspielen von Sicherheitspatches (von Software oder Firmware) verhindert. Solange TC-Technologie Binärdateien über ihre Hashwerte identifiziert, sind verlässliche Managementwerkzeuge erforderlich, die den Administratoren helfen, versiegelte Daten im Falle einer Systemaktualisierung weiterhin verfügbar zu halten. Darüberhinaus müssen diese Werkzeuge in die Managementsoftware von Betriebssystemdistributionen integriert werden, um Erkennung von Modifikationen bei

kritischen Systemkomponenten zu ermöglichen.

Ein weiteres Beispiel von notwenigen Konzepten und Werkzeugen schließt die Entwicklung von angemessenen Lösungen ein, um TPM-geschützte Schlüssel zu verwalten und zu sichern. Im Falle eines Hardwarefehlers (beispielsweise des Motherboards oder des TPM selbst) benötigen Benutzer einen zuverlässigen Mechanismus zur Datenrettung. Da TPM-geschützte Schlüssel das TPM niemals im Klartext verlassen, sollten diese Sicherungskopien durch Software auf Anwendungsebene erstellt und verwaltet werden können.

Eine andere wichtige Erweiterung wäre die Realisierung von Strategien zur Plattform-Migration [10]. Wenn Benutzer ihre Daten zu einer anderen Plattform inkl. eines anderen TPMs (möglicherweise von einem anderen Hersteller) migrieren möchten, müssen ihre TPM-geschützten Daten ebenfalls migriert werden. Das durch die TCG-spezifizierte Migrationgskonzept ist für Hersteller jedoch nur optional und unserem Kenntnisstand nach noch vom keinem TPM-Hersteller umgesetzt worden. Darüberhinaus wird nur eine Migration zu Systemen, die durch ein TPM des gleichen Herstellers gesichert werden, ermöglicht. Um diese Einschränkungen zu umgehen, werden angepasste Werkzeuge benötigt, welche die Datenmigration auf Softwareebene realisieren.

Unserer Kenntnis nach existieren zurzeit keine dieser Werkzeuge, die notwendig dafür wären, eine zuverlässige und sichere Plattform bereitzustellen.

4.3.3. Infrastrukturelle Lösungen

Auf lange Sicht hin müssen „Trusted Computing“-Konzepte so realisiert werden, dass sie die Sicherheit, Zuverlässigkeit und Vertrauenswürdigkeit von bestehenden IT-Infrastrukturen verbessern, indem sie eine Interoperabilität mit bestehenden Lösungen bereitstellen. Mögliche Ansätze sind die Kombination von Technologien zur Virtualisierung, „Trusted Computing“ und Sicherheitskernen, wie von Turaya und OpenTC (siehe Abschnitt 3.2.1) angeboten. Alternativ können bestehende IT-Infrastrukturen durch TC-Technologien erweitert werden, wie am Beispiel der Integrity Measurement Architecture in Abschnitt 4.2 gezeigt wurde.

Diese Betriebssystemarchitekturen lösen jedoch momentan nicht alle Probleme (was sie tatsächlich technisch auch nicht können). Verschiedene offene Fragestellungen existieren in Bezug auf Infrastruktur und Management:

Die TCG spezifiziert beispielsweise sogenannte „Platform Certificates“, die durch den Plattform-Integrator herausgegeben werden, um damit auszudrücken, dass die mit einem TPM ausgestattete Plattform gemäß den TPM Spezifikationen hergestellt wurde. Diese Zertifikate müssen während einer Attestierung oder eines Attestation Identity Key (AIK)-Zertifizierungsprozesses evaluiert werden, um die Vertrauenswürdigkeit dieser Plattform zu

bestimmen. Bis heute bieten Plattformhersteller unseres Wissens nach keine Plattformzertifikate an.

Ein anderer offener Punkt, der bisher nur teilweise durch die „Trusted Network Connect“ (TNC)-Spezifikation [25] betrachtet wurde, ist die Frage, wie die erhebliche Menge an möglichen binären Konfigurationen verwaltet werden kann. In Anbetracht der riesigen Zahl an potentiellen Systemkonfigurationen, Updates, Patches und Compilerversionen ist die Zahl der Konfigurationen schon riesig und wächst exponentiell. Darüberhinaus werden viele OSS-Anwendungen und Betriebssysteme lokal kompiliert, wodurch die Zahl der Konfigurationen abermals vergrößert wird, da viele Compiler die Kompilationszeit und ähnliche dynamische Informationen in die Binärdatei einfügen.

Im Zusammenhang von Attestierung, Bindung und Versiegelung wird oft kritisiert, dass der TCG-Ansatz mehr Informationen über eine Plattform preisgibt, als notwendig. Ein Angreifer kann einerseits binäre Messungen untersuchen, um ein Betriebssystem oder eine Anwendung zu identifizieren und so einen Angriff zu optimieren (Stichwort: *fingerprinting*).

Andererseits erlauben diese binären Werte Herstellern, bestimmte Binärkonfigurationen zu benachteiligen (etwa die Verweigerung der Unterstützung von bestimmten Betriebssystemen oder Anwendungen). Aktuelle Forschungsthemen in den Bereichen *property-based attestation* und *semantic attestation* [15, 8] suchen Lösungen, die nur bestimmte Aspekte einer Plattform attestiert. Die vorgestellten Lösungen sind noch nicht reif für die Anwendung im Massenmarkt.

Die hier beschriebenen Probleme im Bereich des „Trusted Computing“ erstrecken sich über verschiedene Abstraktionsebenen, angefangen von der Konzeption einer Architektur bis hin zur fertigen Lösung und vom Design bis zur aktuellen Implementierung. Trotz der hier aufgezeigten Schwierigkeiten bieten die durch die TCG eingeführten Spezifikationen für das TPM nebst dazugehöriger TSS in Zusammenhang mit den vorgestellten Sicherheitsarchitekturen ein gutes Potential zur Realisierung vertrauenswürdiger Computer-Plattformen. Wie gezeigt wurde, existieren bereits erste Softwarelösungen, die diese Technologie unterstützen. Diese haben zwar noch nicht Produktreife erlangt, tendieren aber in die richtige Richtung.

Anhang

Anhang A. TPM_Unseal

Der folgende Quellcode zeigt eine Implementierung der `tpm_unseal`-Funktion, die verwendet wird, um eine versiegelte Datei mit Hilfe des TPMs wieder zu dechiffrieren. Die benötigte Unseal-Funktion selbst wird in einer Bibliothek von TrouSerS mitgeliefert und durch Einfügen der Header-Datei `tpm_unseal.h` verfügbar gemacht.

```
/*
 * tpm_unseal - A small tool to unseal files using a TPM.
 *
 * Based on:
 * (c) 2007 Gianluca Ramunno (TORSEC group - Politecnico di
 * Torino)
 * This software is released under GPL licence v2.
 */
#include <stdio.h>
#include <tpm_unseal.h>
void usage () {
  printf ("tpm_unseal <sealed_file_name>\n");
  printf (" Upon successful unsealing, the file will be sent to
the standard output\n");
}
int main (int argc, char **argv)
{
  int res, retcode=0, size;
  unsigned char *data;
  if (argc !=2) {
    usage();
    return(2);
  }
  if ((res = tpmUnsealfile( argv[1], &data, &size )) == 0)
    write (1, data, (size_t)size);
  else
    retcode = 1;
  fprintf(stderr, "%s\n", tpmUnsealStrerror(res));
  tpmUnsealShred(data, size);
  return (retcode);
}
```

Kompiliere durch die Eingabe von:

```
gcc -I/usr/include/tpm_tools/ tpm_unseal.c -ltpm_unseal \
-o tpm_unseal.
```

Anhang B. Patchset für TPM Tools

Die folgenden Korrekturen setzen das `TSS_WELL_KNOWN_SECRET` innerhalb der `tpm_sealdata`-Funktion und der `tpm_unsealFile`-Funktion.

Patch für `tpm_sealdata.c`

```
~ $ cat tpm_sealdata.c.diff
--- tpm-tools-1.3.0/src/cmds/tpm_sealdata.c
+++ tpm-tools-1.3.0-patched/src/cmds/tpm_sealdata.c

@@ -118,6 +118,7 @@ int main(int argc, char **argv)
             TSS_KEY_VOLATILE | TSS_KEY_AUTHORIZATION |
             TSS_KEY_NOT_MIGRATABLE;
         TSS_HPOLICY hSrkPolicy;
+  BYTE well_known_secret[TPM_SHA1_160_HASH_LEN] = TSS_WELL_KNOWN_SECRET;

   BIO *bin = NULL, *bdata=NULL, *b64=NULL;

@@ -169,7 +170,8 @@ int main(int argc, char **argv)
   if (policyGet(hSrk, &hSrkPolicy) != TSS_SUCCESS)
                 goto out_close;
-  if (policySetSecret(hSrkPolicy, 0, NULL) != TSS_SUCCESS)
+  if (policySetSecret(hSrkPolicy,TPM_SHA1_160_HASH_LEN,
       well_known_secret)!=TSS_SUCCESS)
+//if (policySetSecret(hSrkPolicy, 0, NULL) != TSS_SUCCESS)
                 goto out_close;
         /* Build an RSA key object that will be created by the TPM
```

Patch für `tpm_unseal.c`

```
~ $ cat tpm_unseal.c.diff
--- tpm-tools-1.3.0/lib/tpm_unseal.c
+++ tpm-tools-1.3.0-patched/lib/tpm_unseal.c

@@ -81,6 +81,7 @@ int tpmUnsealFile( char* fname, unsigned
         TSS_HPOLICY hPolicy;
         UINT32 symKeyLen;
         BYTE *symKey;
+       BYTE well_known_secret[TPM_SHA1_160_HASH_LEN] =
             TSS_WELL_KNOWN_SECRET;
         unsigned char* res_data = NULL;
         int res_size = 0;

@@ -332,7 +333,8 @@ int tpmUnsealFile( char* fname, unsigned
                 goto tss_out;
         }
- if ((rc=Tspi_Policy_SetSecret(hPolicy, TSS_SECRET_MODE_PLAIN, 0, NULL))
+//if ((rc=Tspi_Policy_SetSecret(hPolicy, TSS_SECRET_MODE_PLAIN, 0,
```

```
      NULL))
+ if ((rc=Tspi_Policy_SetSecret(hPolicy, TSS_SECRET_MODE_SHA1,
+ TPM_SHA1_160_HASH_LEN, well_known_secret))
                                          != TSS_SUCCESS) {
                tpm_errno = ETSPIPOLSS;
                goto tss_out;
```

Anhang C. SE-Linux Installation

Die folgenden Schritte sind erforderlich, um ein reguläres Gentoo System in ein gehärtetes SE-Linux umzuwandeln:

1. Kompiliere den SE-Linux erweiterten Kern
2. Wechsle das Systemprofil auf `'enabled hardened support'`
3. Installiere die erforderlichen 'userland utilities'
4. Konfiguriere die Systemumgebung um SE-Linux zu benutzen

Kompilierung von dem SE-Linux erweiterten Kern. Die Kernparameter aus Listing C.1 müssen in Linux konfiguriert werden, um NSA SE-Linux Unterstützung zu aktivieren. Beachten Sie, dass aktuell nur vier Dateisysteme (`ext2, ext3, jfs` und `xfs`) unterstützt werden. Versichern Sie sich zusätzlich, dass die Linuxversion von Adobe Acrobat acroread nicht installiert ist, da diese Anwendung zu einem Problem beim Etikettieren des Dateisystems geführt hat.

Wechseln des Systemprofils. Um die SE-Linux spezifischen Sicherheitseigenschaften zu aktivieren, muss man unter Gentoo zu einem gehärteten Gentoo Profil wechseln. Dabei wird automatisch das USE-flag *selinux* gesetzt (`e.g. USE = selinux`).

Installation notwendiger Werkzeuge. Um SE-Linux systemweit zu benutzen, müssen zusätzliche Bibliotheken, Policies und Werkzuge installiert werden:

```
# emerge sysvinit checkpolicy policycoreutils selinux-base-policy.
```

Danach ist es notwendig, die bereits installierten Anwendungen, welche SE-Linux unterstützen, neu zu kompilieren, damit ihre eigenen Sicherheitsrichtlinien aktiviert werden. Dies geschieht durch den Aufruf von:

```
# emerge --newuse world
```

Konfiguration der Systemumgebung um SE-Linux zu verwenden.
Die Konfiguration von SE-Linux erfolgt in mehreren Schritten:

1. Wähle den Policytyp in `/etc/selinux/config`

```
# This file controls the state of SELinux on the system on
# boot. SELINUX can take one of these three values:
#  enforcing - SELinux security policy is enforced.
#  permissive - SELinux prints warnings instead of enforcing.
```

```
#  disabled - No SELinux policy is loaded.
SELINUX=enforcing

# SELINUXTYPE can take one of these two values:
#   targeted - Only targeted network daemons are protected.
#   strict - Full SELinux protection.
SELINUXTYPE=targeted
```

2. Neustart der Maschine in das installierte SE-Linux

3. Modifiziere `/etc/fstab`

```
none /selinux selinuxfs defaults 0 0
```

4. Etikettiere das Dateisystem mit:

```
# rlpkg -a -r
```

Listing C.2: Wechsel zu einem hardened Gentoo Profil

```
# rm -f /etc/make.profile

x86 ( hardened ) :
# ln -sf /usr/portage/profiles/selinux/2007.0/x86/hardened/ \
  etc/make.profile

AMD64 ( hardened ) :
# ln -sf /usr/portage/profiles/selinux/2007.0/amd64/hardened/ \
  etc/make.profile
```

Listing C.1: SE-Linux Kernkonfiguration

```
cd /usr/src/linux-2.6.22-hardened-r7
make menuconfig

Under Code maturity level options
[*] Prompt for development and / or incomplete code / drivers

Under General setup
[*] Auditing support
[*]     Enable system-call auditing support

Under file systems
<*> Second extended fs support ( If using ext2 )
[*]     Ext2 extended attributes
[ ]        Ext2 POSIX Access Control Lists
[*]        Ext2 Security Labels
<*> Ext3 journalling file system support ( If using ext3 )
[*]     Ext3 extended attributes
[ ]        Ext3 POSIX Access Control Lists
[*]        Ext3 Security labels
<*> JFS filesystem support ( If using JFS )
[ ]     JFS POSIX Access Control Lists
[*]     JFS Security Labels
<*> XFS filesystem support ( If using XFS )
[ ]     Realtime support ( EXPERIMENTAL )
[ ]     Quota support
[ ]     ACL support
[*]     Security Labels

Under file systems -> Pseudo file systems
[ ] /dev file system support ( EXPERIMENTAL )
[*]     /dev/pts Extended Attributes
[*]        /dev/pts Security Labels
[*] Virtual memory file system support ( former shm fs )
[*]     tmpfs Extended Attributes
[*]        tmpfs Security Labels

Under Security options
[*] Enable different security models
[*]     Socket and Networking Security Hooks
<*>     Default Linux Capabilities
[*] NSA SELinux Support
[ ]     NSA SELinux boot parameter
[ ]     NSA SELinux runtime disable
[*]     NSA SELinux Development Support
[ ]     NSA SELinux AVC Statistics
(1)     NSA SELinux checkreqprot default value
[ ]     NSA SELinux enable new secmark network controls by default
[ ]     NSA SELinux maximum supported policy format version
```

Anhang D. Creative Commons Lizenz by ND

CREATIVE COMMONS IST KEINE RECHTSANWALTSKANZLEI UND LEISTET KEINE RECHTSBERATUNG. DIE BEREITSTELLUNG DIESER LIZENZ FÜHRT ZU KEINEM MANDATSVERHÄLTNIS. CREATIVE COMMONS STELLT DIESE INFORMATIONEN OHNE GEWÄHR ZUR VERFÜGUNG. CREATIVE COMMONS ÜBERNIMMT KEINE GEWÄHRLEISTUNG FÜR DIE GELIEFERTEN INFORMATIONEN UND SCHLIEßT DIE HAftUNG FÜR SCHÄDEN AUS, DIE SICH AUS DEREN GEBRAUCH ERGEBEN.

Lizenz: Namensnennung – Keine Bearbeitung 3.0 Deutschland

DER GEGENSTAND DIESER LIZENZ (WIE UNTER "SCHUTZGEGENSTAND" DEfiNIERT) WIRD UNTER DEN BEDINGUNGEN DIESER CREATIVE COMMONS PUBLIC LICENSE ("CCPL", "LIZENZ" ODER "LIZENZVERTRAG") ZUR VERFÜGUNG GESTELLT. DER SCHUTZGEGENSTAND IST DURCH DAS URHEBERRECHT UND/ODER ANDERE GESETZE GESCHÜTZT. JEDE FORM DER NUTZUNG DES SCHUTZGEGENSTANDES, DIE NICHT AUFGRUND DIESER LIZENZ ODER DURCH GESETZE GESTATTET IST, IST UNZULÄSSIG.

DURCH DIE AUSÜBUNG EINES DURCH DIESE LIZENZ GEWÄHRTEN RECHTS AN DEM SCHUTZGEGENSTAND ERKLÄREN SIE SICH MIT DEN LIZENZBEDINGUNGEN RECHTSVERBINDLICH EINVERSTANDEN. SOWEIT DIESE LIZENZ ALS LIZENZVERTRAG ANZUSEHEN IST, GEWÄHRT IHNEN DER LIZENZGEBER DIE IN DER LIZENZ GENANNTEN RECHTE UNENTGELTLICH UND IM AUSTAUSCH DAFÜR, DASS SIE DAS GEBUNDENSEIN AN DIE LIZENZBEDINGUNGEN AKZEPTIEREN.

1. Definitionen

a. Der Begriff "**Abwandlung**" im Sinne dieser Lizenz bezeichnet das Ergebnis jeglicher Art von Veränderung des Schutzgegenstandes, solange die eigenpersönlichen Züge des Schutzgegenstandes darin nicht verblassen und daran eigene Schutzrechte entstehen. Das kann insbesondere eine Bearbeitung, Umgestaltung, Änderung, Anpassung, Übersetzung oder Heranziehung des Schutzgegenstandes zur Vertonung von Laufbildern sein. Nicht als Abwandlung des Schutzgegenstandes gelten seine Aufnahme in eine Sammlung oder ein Sammelwerk und die freie Benutzung des Schutzgegenstandes.
b. Der Begriff "**Sammelwerk**" im Sinne dieser Lizenz meint eine Zusammenstellung von literarischen, künstlerischen oder wissenschaftlichen Inhalten, sofern diese Zusammenstellung aufgrund von Auswahl und Anordnung der darin enthaltenen selbständigen Elemente eine geistige Schöpfung darstellt, unabhängig davon, ob die Elemente systematisch oder methodisch angelegt und dadurch einzeln zugänglich sind oder nicht.
c. "**Verbreiten**" im Sinne dieser Lizenz bedeutet, den Schutzgegenstand im Original oder in Form von Vervielfältigungsstücken, mithin in körperlich fixierter Form der Öffentlichkeit anzubieten oder in Verkehr zu bringen.
d. Der "**Lizenzgeber**" im Sinne dieser Lizenz ist diejenige natürliche oder juristische Person

oder Gruppe, die den Schutzgegenstand unter den Bedingungen dieser Lizenz anbietet und insoweit als Rechteinhaberin auftritt.

e. "**Rechteinhaber**" im Sinne dieser Lizenz ist der Urheber des Schutzgegenstandes oder jede andere natürliche oder juristische Person oder Gruppe von Personen, die am Schutzgegenstand ein Immaterialgüterrecht erlangt hat, welches die in Abschnitt 3 genannten Handlungen erfasst und bei dem eine Einräumung von Nutzungsrechten oder eine Weiterübertragung an Dritte möglich ist.
f. Der Begriff "**Schutzgegenstand**" bezeichnet in dieser Lizenz den literarischen, künstlerischen oder wissenschaftlichen Inhalt, der unter den Bedingungen dieser Lizenz angeboten wird. Das kann insbesondere eine persönliche geistige Schöpfung jeglicher Art, ein Werk der kleinen Münze, ein nachgelassenes Werk oder auch ein Lichtbild oder anderes Objekt eines verwandten Schutzrechts sein, unabhängig von der Art seiner Fixierung und unabhängig davon, auf welche Weise jeweils eine Wahrnehmung erfolgen kann, gleichviel ob in analoger oder digitaler Form. Soweit Datenbanken oder Zusammenstellungen von Daten einen immaterialgüterrechtlichen Schutz eigener Art genießen, unterfallen auch sie dem Begriff "Schutzgegenstand" im Sinne dieser Lizenz.
g. Mit "**Sie**" bzw. "**Ihnen**" ist die natürliche oder juristische Person gemeint, die in dieser Lizenz im Abschnitt 3 genannte Nutzungen des Schutzgegenstandes vornimmt und zuvor in Hinblick auf den Schutzgegenstand nicht gegen Bedingungen dieser Lizenz verstoßen oder aber die ausdrückliche Erlaubnis des Lizenzgebers erhalten hat, die durch diese Lizenz gewährten Nutzungsrechte trotz eines vorherigen Verstoßes auszuüben.
h. Unter "**Öffentlich Zeigen**" im Sinne dieser Lizenz sind Veröffentlichungen und Präsentationen des Schutzgegenstandes zu verstehen, die für eine Mehrzahl von Mitgliedern der Öffentlichkeit bestimmt sind und in unkörperlicher Form mittels öffentlicher Wiedergabe in Form von Vortrag, Aufführung, Vorführung, Darbietung, Sendung, Weitersendung, zeit- und ortsunabhängiger Zugänglichmachung oder in körperlicher Form mittels Ausstellung erfolgen, unabhängig von bestimmten Veranstaltungen und unabhängig von den zum Einsatz kommenden Techniken und Verfahren, einschließlich drahtgebundener oder drahtloser Mittel und Einstellen in das Internet.
i. "**Vervielfältigen**" im Sinne dieser Lizenz bedeutet, mittels beliebiger Verfahren Vervielfältigungsstücke des Schutzgegenstandes herzustellen, insbesondere durch Ton- oder Bildaufzeichnungen, und umfasst auch den Vorgang, erstmals körperliche Fixierungen des Schutzgegenstandes sowie Vervielfältigungsstücke dieser Fixierungen anzufertigen, sowie die Übertragung des Schutzgegenstandes auf einen Bild- oder Tonträger oder auf ein anderes elektronisches Medium, gleichviel ob in digitaler oder analoger Form.

2. Schranken des Immaterialgüterrechts

Diese Lizenz ist in keiner Weise darauf gerichtet, Befugnisse zur Nutzung des Schutzgegenstandes zu vermindern, zu beschränken oder zu vereiteln, die Ihnen aufgrund der Schranken des Urheberrechts oder anderer Rechtsnormen bereits ohne Weiteres zustehen oder sich aus dem Fehlen eines immaterialgüterrechtlichen Schutzes ergeben.

3. Einräumung von Nutzungsrechten

Unter den Bedingungen dieser Lizenz räumt Ihnen der Lizenzgeber - unbeschadet unverzichtbarer Rechte und vorbehaltlich des Abschnitts 3.c) - das vergütungsfreie, räumlich und zeitlich (für die Dauer des Schutzrechts am Schutzgegenstand) unbeschränkte einfache Recht ein, den Schutzgegenstand auf die folgenden Arten und Weisen zu nutzen ("unentgeltlich eingeräumtes

einfaches Nutzungsrecht für jedermann"):

a. Den Schutzgegenstand in beliebiger Form und Menge zu vervielfältigen, ihn in Sammelwerke zu integrieren und ihn als Teil solcher Sammelwerke zu vervielfältigen;
b. den Schutzgegenstand, allein oder in Sammelwerke aufgenommen, öffentlich zu zeigen und zu verbreiten.
c. Bezüglich Vergütung für die Nutzung des Schutzgegenstandes gilt Folgendes:
    i. **Unverzichtbare gesetzliche Vergütungsansprüche**: Soweit unverzichtbare Vergütungsansprüche im Gegenzug für gesetzliche Lizenzen vorgesehen oder Pauschalabgabensysteme (zum Beispiel für Leermedien) vorhanden sind, behält sich der Lizenzgeber das ausschließliche Recht vor, die entsprechende Vergütung einzuziehen für jede Ausübung eines Rechts aus dieser Lizenz durch Sie.
    ii. **Vergütung bei Zwangslizenzen**: Sofern Zwangslizenzen außerhalb dieser Lizenz vorgesehen sind und zustande kommen, verzichtet der Lizenzgeber für alle Fälle einer lizenzgerechten Nutzung des Schutzgegenstandes durch Sie auf jegliche Vergütung.
    iii. **Vergütung in sonstigen Fällen**: Bezüglich lizenzgerechter Nutzung des Schutzgegenstandes durch Sie, die nicht unter die beiden vorherigen Abschnitte (i) und (ii) fällt, verzichtet der Lizenzgeber auf jegliche Vergütung, unabhängig davon, ob eine Einziehung der Vergütung durch ihn selbst oder nur durch eine Verwertungsgesellschaft möglich wäre.

Das vorgenannte Nutzungsrecht wird für alle bekannten sowie für alle noch nicht bekannten Nutzungsarten eingeräumt. Es beinhaltet auch das Recht, solche Änderungen am Schutzgegenstand vorzunehmen, die für bestimmte nach dieser Lizenz zulässige Nutzungen technisch erforderlich sind. Weitergehende Änderungen oder Abwandlungen sind jedoch untersagt. Alle sonstigen Rechte, die über diesen Abschnitt hinaus nicht ausdrücklich durch den Lizenzgeber eingeräumt werden, bleiben diesem allein vorbehalten. Soweit Datenbanken oder Zusammenstellungen von Daten Schutzgegenstand dieser Lizenz oder Teil dessen sind und einen immaterialgüterrechtlichen Schutz eigener Art genießen, verzichtet der Lizenzgeber auf sämtliche aus diesem Schutz resultierenden Rechte.

4. Bedingungen

Die Einräumung des Nutzungsrechts gemäß Abschnitt 3 dieser Lizenz erfolgt ausdrücklich nur unter den folgenden Bedingungen:

a. Sie dürfen den Schutzgegenstand ausschließlich unter den Bedingungen dieser Lizenz verbreiten oder öffentlich zeigen. Sie müssen dabei stets eine Kopie dieser Lizenz oder deren vollständige Internetadresse in Form des Uniform-Resource-Identifier (URI) beifügen. Sie dürfen keine Vertrags- oder Nutzungsbedingungen anbieten oder fordern, die die Bedingungen dieser Lizenz oder die durch diese Lizenz gewährten Rechte beschränken. Sie dürfen den Schutzgegenstand nicht unterlizenzieren. Bei jeder Kopie des Schutzgegenstandes, die Sie verbreiten oder öffentlich zeigen, müssen Sie alle Hinweise unverändert lassen, die auf diese Lizenz und den Haftungsausschluss hinweisen. Wenn Sie den Schutzgegenstand verbreiten oder öffentlich zeigen, dürfen Sie (in Bezug auf den Schutzgegenstand) keine technischen Maßnahmen ergreifen, die den Nutzer des Schutzgegenstandes in der Ausübung der ihm durch diese Lizenz gewährten Rechte behindern können. Dieser Abschnitt 4.a) gilt auch für den Fall, dass der Schutzgegenstand einen Bestandteil eines Sammelwerkes bildet, was jedoch nicht bedeutet, dass das

Sammelwerk insgesamt dieser Lizenz unterstellt werden muss. Sofern Sie ein Sammelwerk erstellen, müssen Sie auf die Mitteilung eines Lizenzgebers hin aus dem Sammelwerk die in Abschnitt 4.b) aufgezählten Hinweise entfernen.

b. Die Verbreitung und das öffentliche Zeigen des Schutzgegenstandes oder ihn enthaltender Sammelwerke ist Ihnen nur unter der Bedingung gestattet, dass Sie, vorbehaltlich etwaiger Mitteilungen im Sinne von Abschnitt 4.a), alle dazu gehörenden Rechtevermerke unberührt lassen. Sie sind verpflichtet, die Rechteinhaberschaft in einer der Nutzung entsprechenden, angemessenen Form anzuerkennen, indem Sie - soweit bekannt - Folgendes angeben:

    i. Den Namen (oder das Pseudonym, falls ein solches verwendet wird) des Rechteinhabers und / oder, falls der Lizenzgeber im Rechtevermerk, in den Nutzungsbedingungen oder auf andere angemessene Weise eine Zuschreibung an Dritte vorgenommen hat (z.B. an eine Stiftung, ein Verlagshaus oder eine Zeitung) ("Zuschreibungsempfänger"), Namen bzw. Bezeichnung dieses oder dieser Dritten;
    ii. den Titel des Inhaltes;
    iii. in einer praktikablen Form den Uniform-Resource-Identifier (URI, z.B. Internetadresse), den der Lizenzgeber zum Schutzgegenstand angegeben hat, es sei denn, dieser URI verweist nicht auf den Rechtevermerk oder die Lizenzinformationen zum Schutzgegenstand.

    Die nach diesem Abschnitt 4.b) erforderlichen Angaben können in jeder angemessenen Form gemacht werden; im Falle eines Sammelwerkes müssen diese Angaben das Minimum darstellen und bei gemeinsamer Nennung mehrerer Rechteinhaber dergestalt erfolgen, dass sie zumindest ebenso hervorgehoben sind wie die Hinweise auf die übrigen Rechteinhaber. Die Angaben nach diesem Abschnitt dürfen Sie ausschließlich zur Angabe der Rechteinhaberschaft in der oben bezeichneten Weise verwenden. Durch die Ausübung Ihrer Rechte aus dieser Lizenz dürfen Sie ohne eine vorherige, separat und schriftlich vorliegende Zustimmung des Lizenzgebers und / oder des Zuschreibungsempfängers weder explizit noch implizit irgendeine Verbindung zum Lizenzgeber oder Zuschreibungsempfänger und ebenso wenig eine Unterstützung oder Billigung durch ihn andeuten.

c. Die oben unter 4.a) und b) genannten Einschränkungen gelten nicht für solche Teile des Schutzgegenstandes, die allein deshalb unter den Schutzgegenstandsbegriff fallen, weil sie als Datenbanken oder Zusammenstellungen von Daten einen immaterialgüterrechtlichen Schutz eigener Art genießen.
d. Persönlichkeitsrechte bleiben - soweit sie bestehen - von dieser Lizenz unberührt.

5. Gewährleistung

SOFERN KEINE ANDERS LAUTENDE, SCHRIftLICHE VEREINBARUNG ZWISCHEN DEM LIZENZGEBER UND IHNEN GESCHLOSSEN WURDE UND SOWEIT MÄNGEL NICHT ARGLISTIG VERSCHWIEGEN WURDEN, BIETET DER LIZENZGEBER DEN SCHUTZGEGENSTAND UND DIE EINRÄUMUNG VON RECHTEN UNTER AUSSCHLUSS JEGLICHER GEWÄHRLEISTUNG AN UND ÜBERNIMMT WEDER AUSDRÜCKLICH NOCH KONKLUDENT GARANTIEN IRGENDEINER ART. DIES UMFASST INSBESONDERE DAS FREISEIN VON SACH- UND RECHTSMÄNGELN, UNABHÄNGIG VON DEREN ERKENNBARKEIT FÜR DEN LIZENZGEBER, DIE VERKEHRSFÄHIGKEIT DES SCHUTZGEGENSTANDES, SEINE VERWENDBARKEIT FÜR EINEN BESTIMMTEN ZWECK SOWIE DIE KORREKTHEIT VON BESCHREIBUNGEN. DIESE

GEWÄHRLEISTUNGSBESCHRÄNKUNG GILT NICHT, SOWEIT MÄNGEL ZU SCHÄDEN DER IN ABSCHNITT 6 BEZEICHNETEN ART FÜHREN UND AUF SEITEN DES LIZENZGEBERS DAS JEWEILS GENANNTE VERSCHULDEN BZW. VERTRETENMÜSSEN EBENFALLS VORLIEGT.

6. Haftungsbeschränkung

DER LIZENZGEBER HAftET IHNEN GEGENÜBER IN BEZUG AUF SCHÄDEN AUS DER VERLETZUNG DES LEBENS, DES KÖRPERS ODER DER GESUNDHEIT NUR, SOFERN IHM WENIGSTENS FAHRLÄSSIGKEIT VORZUWERFEN IST, FÜR SONSTIGE SCHÄDEN NUR BEI GROBER FAHRLÄSSIGKEIT ODER VORSATZ, UND ÜBERNIMMT DARÜBER HINAUS KEINERLEI FREIWILLIGE HAftUNG.

7. Erlöschen

a. Diese Lizenz und die durch sie eingeräumten Nutzungsrechte erlöschen mit Wirkung für die Zukunft im Falle eines Verstoßes gegen die Lizenzbedingungen durch Sie, ohne dass es dazu der Kenntnis des Lizenzgebers vom Verstoß oder einer weiteren Handlung einer der Vertragsparteien bedarf. Mit natürlichen oder juristischen Personen, die den Schutzgegenstand enthaltende Sammelwerke unter den Bedingungen dieser Lizenz von Ihnen erhalten haben, bestehen nachträglich entstandene Lizenzbeziehungen jedoch solange weiter, wie die genannten Personen sich ihrerseits an sämtliche Lizenzbedingungen halten. Darüber hinaus gelten die Ziffern 1, 2, 5, 6, 7, und 8 auch nach einem Erlöschen dieser Lizenz fort.
b. Vorbehaltlich der oben genannten Bedingungen gilt diese Lizenz unbefristet bis der rechtliche Schutz für den Schutzgegenstand ausläuft. Davon abgesehen behält der Lizenzgeber das Recht, den Schutzgegenstand unter anderen Lizenzbedingungen anzubieten oder die eigene Weitergabe des Schutzgegenstandes jederzeit einzustellen, solange die Ausübung dieses Rechts nicht einer Kündigung oder einem Widerruf dieser Lizenz (oder irgendeiner Weiterlizenzierung, die auf Grundlage dieser Lizenz bereits erfolgt ist bzw. zukünftig noch erfolgen muss) dient und diese Lizenz unter Berücksichtigung der oben zum Erlöschen genannten Bedingungen vollumfänglich wirksam bleibt.

8. Sonstige Bestimmungen

a. Jedes Mal wenn Sie den Schutzgegenstand für sich genommen oder als Teil eines Sammelwerkes verbreiten oder öffentlich zeigen, bietet der Lizenzgeber dem Empfänger eine Lizenz zu den gleichen Bedingungen und im gleichen Umfang an, wie Ihnen in Form dieser Lizenz.
b. Sollte eine Bestimmung dieser Lizenz unwirksam sein, so bleibt davon die Wirksamkeit der Lizenz im Übrigen unberührt.
c. Keine Bestimmung dieser Lizenz soll als abbedungen und kein Verstoß gegen sie als zulässig gelten, solange die von dem Verzicht oder von dem Verstoß betroffene Seite nicht schriftlich zugestimmt hat.
d. Diese Lizenz (zusammen mit in ihr ausdrücklich vorgesehenen Erlaubnissen, Mitteilungen und Zustimmungen, soweit diese tatsächlich vorliegen) stellt die vollständige Vereinbarung zwischen dem Lizenzgeber und Ihnen in Bezug auf den Schutzgegenstand dar. Es bestehen keine Abreden, Vereinbarungen oder Erklärungen in Bezug auf den Schutzgegenstand, die in dieser Lizenz nicht genannt sind. Rechtsgeschäftliche Änderungen des Verhältnisses zwischen dem Lizenzgeber und Ihnen sind nur über Modifikationen dieser Lizenz möglich. Der Lizenzgeber ist an etwaige zusätzliche, einseitig durch Sie übermittelte Bestimmungen

nicht gebunden. Diese Lizenz kann nur durch schriftliche Vereinbarung zwischen Ihnen und dem Lizenzgeber modifiziert werden. Derlei Modifikationen wirken ausschließlich zwischen dem Lizenzgeber und Ihnen und wirken sich nicht auf die Dritten gemäß Ziffern 8.a) angeboteten Lizenzen aus.

e. Sofern zwischen Ihnen und dem Lizenzgeber keine anderweitige Vereinbarung getroffen wurde und soweit Wahlfreiheit besteht, findet auf diesen Lizenzvertrag das Recht der Bundesrepublik Deutschland Anwendung.

Creative Commons Notice

Creative Commons ist nicht Partei dieser Lizenz und übernimmt keinerlei Gewähr oder dergleichen in Bezug auf den Schutzgegenstand. Creative Commons haftet Ihnen oder einer anderen Partei unter keinem rechtlichen Gesichtspunkt für irgendwelche Schäden, die - abstrakt oder konkret, zufällig oder vorhersehbar - im Zusammenhang mit dieser Lizenz entstehen. Unbeschadet der vorangegangen beiden Sätze, hat Creative Commons alle Rechte und Pflichten eines Lizenzgebers, wenn es sich ausdrücklich als Lizenzgeber im Sinne dieser Lizenz bezeichnet.

Creative Commons gewährt den Parteien nur insoweit das Recht, das Logo und die Marke "Creative Commons" zu nutzen, als dies notwendig ist, um der Öffentlichkeit gegenüber kenntlich zu machen, dass der Schutzgegenstand unter einer CCPL steht. Ein darüber hinaus gehender Gebrauch der Marke "Creative Commons" oder einer verwandten Marke oder eines verwandten Logos bedarf der vorherigen schriftlichen Zustimmung von Creative Commons. Jeder erlaubte Gebrauch richtet sich nach der Creative Commons Marken-Nutzungs-Richtlinie in der jeweils aktuellen Fassung, die von Zeit zu Zeit auf der Website veröffentlicht oder auf andere Weise auf Anfrage zugänglich gemacht wird. Zur Klarstellung: Die genannten Einschränkungen der Markennutzung sind nicht Bestandteil dieser Lizenz.

Creative Commons kann kontaktiert werden über http://creativecommons.org/.

Literaturverzeichnis

[1] Ammar Alkassar, Michael Scheibel, Ahmad-Reza Sadeghi, Christian Stüble, and Marcel Winandy. Security architecture for device encryption and VPN. In Sachar Paulus, Norbert Pohlmann, and Helmut Reimer, editors, Information Security Solutions Europe (ISSE 2006), pages 54–63. Vieweg Verlag, 2006.

[2] AMD. AMD64 virtualization codenamed “Pacifica” technology - secure virtual machine architecture reference manual. Technical Report Publication Number 33047, Revision 3.01, AMD, May 2005.

[3] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Adrew Warfield. Xen and the art of virtualization. In Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP’03), Bolton Landing, NY, USA, October 2003. ACM.

[4] Stefan Berger, Ramon Caceres, Kenneth A. Goldman, Ronald Perez, Reiner Sailer, and Leendert van Doorn. vTPM: Virtualizing the Trusted Platform Module. In Proceedings of the 15th USENIX Security Symposium, pages 305–320. USENIX, August 2006.

[5] Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. Rz 3540, IBM Research, March 2004.

[6] Trusted Computing Group. TCG software stack specification http://trustedcomputinggroup.org, August 2003. Version 1.1.

[7] Trusted Computing Group. TCG software stack specification. http://trustedcomputinggroup.org, January 2006. Version 1.2.

[8] Vivek Haldar, Deepak Chandra, and Michael Franz. Semantic remote attestation: A virtual machine directed approach to trusted computing. In USENIX Virtual Machine Research and Technology Symposium, May 2004. also Technical Report No. 03-20, School of Information and Computer Science, University of California, Irvine; October 2003.

[9] IBM Inc. The TrouSerS webpage. http://trousers.sourceforge.net/, July 2007.

[10] Ulrich Kühn, Klaus Kursawe, Stefan Lucks, Ahmad-Reza Sadeghi, and Christian Stüble. Secure data management in trusted computing. In Josyula R. Rao and Berk Sunar, editors, Cryptographic Hardware and Embedded Systems - CHES 2005, volume 3659 of Lecture Notes in Computer Science, pages 324–338. Springer-Verlag, Berlin Germany, 2005.

[11] Ulrich Kühn, Marcel Selhorst, and Christian Stüble. Realizing property-based attestation and sealing with commonly available hard- and software. In Proceedings of the 2nd ACM Workshop on Scalable Trusted Computing (STC'07). ACM Press, 2007.

[12] Aravind Menon, Alan L. Cox, and Willy Zwaenepoel. Optimizing network virtualization in Xen. In USENIX Annual Technical Conference, Boston, MA, 2006.

[13] The OpenSSL Project. The OpenSSL webpage. http://www.openssl.org/, July 2007.

[14] The TPM Manager project. The TPM Manager website. http://www.sourceforge.net/projects/tpmmanager/, July 2007.

[15] Ahmad-Reza Sadeghi and Christian St¨ble. Property-based attestation for computing platforms: Caring about properties, not mechanisms. In The 2004 New Security Paradigms Workshop, Virginia Beach, VA, USA, September 2004. ACM SIGSAC, ACM Press.

[16] Ahmad-Reza Sadeghi, Christian Stüble, and Norbert Pohlmann. European multilateral secure computing base - open trusted computing for you and me. Datenschutz und Datensicherheit DuD, Verlag Friedrich Vieweg & Sohn, Wiesbaden, 28(9):548–554, 2004.

[17] Reiner Sailer, Xiaolan Zhang, Trent Jaeger, and Leendert van Doorn. Design and implementation of a TCG-based integrity measurement architecture. 13th Usenix Security Symposium, San Diego, California, pages 223–238, August 2004.

[18] Stephen Smalley. Configuring the SELinux policy. Report #02-007, NAI Labs, February 2002. Revised April 2002.

[19] Stephen Smalley, Chris Vance, and Wayne Salamon. Implementing SELinux as a Linux security module. Report #01-043, NAI Labs, December 2001. Revised May 2002.

[20] Ray Spencer, Stephen Smalley, Peter Loscocco, Mike Hibler, David Andersen, and Jay Lepreau. The flask security architecture: System support for diverse security policies. In Proceedings of the 8th USENIX Security Symposium, pages 123–139, Washington, D.C., USA, August 1999. USENIX.

[21] Christian Stüble and Anoosheh Zaerin. TPM Manager – Requirements, Design, and Implementation. Technical report, Sirrix AG security technologies and Ruhr-University Bochum, 2006.

[22] Trusted Computing Group. TCG PC-client specific implementation for conventional BIOS version 1.20 FINAL. Technical report, Trusted Computing Group, Incorporated, July 2005.

[23] Trusted Computing Group. TPM interface specification. Specification Version 1.0, Trusted Computing Group, 2005.

[24] Trusted Computing Group. TPM main specification. Main Specification Version 1.2 rev. 85, Trusted Computing Group, February 2005.

[25] Trusted Computing Group. Trusted network connect. Specification Version 1.2,2007.

[26] Trusted Computing Platform Alliance (TCPA). TCPA PC-specific implementation specification, September 2001. Version 1.00.

[27] Trusted Computing Platform Alliance (TCPA). Main specification, February 2002. Version 1.1b.

[28] Eric A. Young and Tim J. Hudson. The SSLeay project. http://www.columbia.edu/~ariel/ssleay/, July 2007.

Abkürzungsverzeichnis

AIK	Attestation Identity Key
API	Application Programming Interface
BIOS	Basic Input Output System
CHS	Cylinder Head Sector
CRTM	Core Root of Trust for Measurement
DAA	Direct Anonymous Attestation
D-RTM	Dynamic Root of Trust for Measurement
EK	Endorsement Key
FSS	Free Software Foundation
GNU	GNU's Not Unix
GPL	GNU General Public License
GRUB	GRand Unified Bootloader
GUI	Graphical User Interface
IMA	Integrity Measurement Architecture
IPC	Inter-Process Communication
LBA	Logical Block Addressing
MAC	Mandatory Access Control
MBR	Master Boot Record
OSI	Open Source Initiative
OSLO	Open Secure Loader
OSS	Open Source Software
PCR	Platform Configuration Register
PrivacyCA	Privacy Certification Authority
RTM	Root of Trust for Measurement
SLOC	Source Lines Of Code
SRK	Storage Root Key
S-RTM	Static Root of Trust for Measurement
TC	Trusted Computing
TCG	Trusted Computing Group
TCS	TSS Core Service
TDD	TPM Device Driver

TDDL	TCG Device Driver Library
TDDLI	TCG Device Driver Library Interface
TIS	TPM Interface Specification
TNC	Trusted Network Connect
TPM	Trusted Platform Module
TSP	TSS Service Provider
TSPI	TSS Service Provider Interface
TSS	TCG Software Stack
UML	Unified Modeling Language
VM	Virtual Machine
VMM	Virtual Machine Monitor

Glossar

Attestation Identity Key

Ein nicht migrierbarer Schlüssel (Non-Migratable Key), der lokal von einem TPM erzeugt wird und Pseudonymität oder Anonymität der durch das TPM gesicherten Plattform erzielt. Der öffentliche Teil eines AIK wird von einer Privacy Certification Authority (PrivacyCA) zertifiziert, die darlegt, dass dieser Signaturschlüssel tatsächlich von einem sicheren TPM kontrolliert wird. Um das Problem zu umgehen, dass die PrivacyCA dadurch Transaktionen einer bestimmten Plattform zuordnen kann, ist in der Version 1.2 der TCG-Spezifikation ein kryptographisches Protokoll namens Direct Anonymous Attestation (DAA) [5] definiert, welches die Notwendigkeit einer PrivacyCA vermeidet.

Basic Input Output System

Der Code, der auf einer PC-Plattform den Speicher und Hardwaregeräte initialisiert.

Core Root of Trust for Measurement

Eine von der TCG spezifizierte PC-Komponente, die das BIOS misst, bevor dieses ausgeführt wird.

Cylinder Head Sector

Eine frühe Methode, um jedem physikalischen Datenblock eines Festplattenlaufwerks eine Adresse zuzuweisen.

Direct Anonymous Attestation

Ein kryptographisches Protokoll, das im Zusammenhang mit der TCG- Spezifikation entwickelt wurde [5], um das Problem zu umgehen, dass eine dritte Partei Transaktionen zu eine bestimmte Plattform zuordnen kann; vermeidet die Notwendigkeit einer PrivacyCA durch Verwendung eines Zero Knowledge-Protokolls.

Dynamic Root of Trust for Measurement

Das Dynamic Root of Trust for Measurement (D-RTM) ist ein Root of Trust for Measurement (RTM), welches von Intels Hardware-Erweiterung TXT oder AMDs Presidio unterstützt wird. Hierbei läuft dynamisch ladbarer Programmcode in einer isolierten Umgebung des Prozessors, so dass dieser die Basis der Vertrauenskette bildet. Virtuelle Maschinen erhalten dadurch jeweils ein eigenes RTM. Die Integritätswerte eines D-RTM werden in den PCRs 17 - 22 abgelegt.

Endorsement Key

Ein asymmetrisches 2048-Bit RSA-Schlüsselpaar, welches eindeutig einem TPM zugeordnet ist. Der EK verbleibt permanent im TPM und kann genutzt werden, um ein TPM und seine Plattform zu authentifizieren.

Free Software Foundation

siehe http://www.fsf.org/

Freie Software

Ist synonym zu Open Source Software (OSS). Die Lizenz einer Freien / Open Source Software (F/OSS) muss den Kriterien der Free Software Foundation (siehe http://www.fsf.org/licensing/essays/free-sw.html) oder den äquivalenten Kriterien der Open Source Initiative (siehe http://opensource.org/docs/osd) genügen.

GNU General Public License

Die am weitesten verbreitete Lizenz für Freie Software.

GNU's Not Unix

Das GNU-Projekt wurde 1984 ins Leben gerufen, mit dem Ziel ein vollständiges Unix-artiges Betriebssystem das auf freier Software basiert zu entwickeln. Varianten des GNU-Betriebssystems, welche einen Kern namens Linux verwenden, werden heutzutage viel verwendet; obwohl diese Systeme häufig als „Linux" bezeichnet werden, ist die genauere Bezeichnung „GNU/Linux- System". GNU ist ein rekursives Akronym für „GNU's Not Unix"; es wird „guh-noo" ausgesprochen, ähnlich wie „canoe". Für weitere Informationen siehe http://www.gnu.org/.

GRand Unified Bootloader

Ein Bootloader-Paket des GNU's Not Unix (GNU)-Projekts, welches die Multiboot-Spezifikation implementiert, die es Nutzern erlaubt, mehrere verschiedene Betriebssysteme gleichzeitig auf ihrem Computer zu installieren.

Hypervisor

Siehe Virtual Machine Monitor.

Integrity Measurement Architecture

Eine von IBM entwickelte Architektur, die es erlaubt, repräsentative Informationen aus einem Linux Software Stack zu generieren. Dadurch kann ein entfernter Teilnehmer anhand dieser Informationen die Integrität des Linux Systems zur Laufzeit verifizieren.

Linux

Ein Synonym für den Linux Kernel; wird normalerweise immer mit der Versionsnummer angegeben. Beispiel: Linux-2.6.22.1. Oftmals wird mit Linux in der Umgangsprache das „GNU Linux Betriebssystem" gemeint.

Logical Block Addressing

Ein weit verbreitetes Adressierungsverfahren für blockorientiere Datenträger wie Festplatten oder USB-Sticks. Mit Hilfe dieses Verfahrens ist es möglich lesend oder schreibend auf die Daten des Datenträgers zuzugreifen.

Mandatory Access Control

Ein Konzept für die Kontrolle und Steuerung von Zugriffsrechten. Die Entscheidung über Zugriffsberechtigungen werden nicht nur auf der Basis der Identität des Akteurs (Benutzers, Prozesses) und des Objektes (Die Ressource auf welche Zugegriffen werden möchte) gefällt, sondern aufgrund zusätzlicher Regeln und Eigenschaften (wie Kategorisierungen, Etiketten und Code-Wörtern). In den „Trusted Computer System Evaluation Criteria" (oft auch als Orange Book des Department of Defense (DoD) bezeichnet) ist der Begriff wie folgt definiert: Mandatory Access Control Beschränken den Zugriff auf Objekten zum einen anhand der Sensitivität der Daten die in dem Objekt enthalten sind - welche durch ein Label repräsentiert wird - und zum anderen durch formale Berechtigung von Subjekten, um an Informationen mit der gleichen Sensitivität zu gelangen.

Non-Migratable Key

Im Gegensatz zu einem migrierbaren Schlüssel, garantiert ein nicht-migrierbarer Schlüssel den Verbleib in einem TPM. Das TPM kann zusätzlich ein Zertifikat erzeugen, welches bescheinigt, dass es sich um einen nicht-migrierbaren Schlüssel handelt.

Open Secure Loader

Ein sicherer Bootloader von der TU Dresden, der AMDs SKINIT Instruktionen einsetzt.

Open Source Initiative

siehe http://opensource.org/

Open Source Software

Ist synonym zu Freier Software (OSS). Die Lizenz einer Freien / Open Source Software (F/OSS) muss den Kriterien der Free Software Foundation (siehe http://www.fsf.org/licensing/essays/free-sw.html) oder den äquivalenten Kriterien der Open Source Initiative (siehe http://opensource.org/docs/osd) genügen.

Platform Configuration Register

Ein TPM-Register, in dem die Konfigurationen von Softwarekomponenten gespeichert werden.

Privacy Certification Authority

Eine Trusted Third Party (TTP) die angibt, ob ein AIK auch wirklich von einem TPM kontrolliert wird.

Root of Trust for Measurement

Das Root of Trust for Measurement (bei PCs auch CRTM) bildet die Basis der Integritätsmessungen einer Plattform. Das RTM ist das erste Element der Vertrauenskette, welches über den Bootloader und den Kernel bis zu Anwendungen des Betriebssystems reichen kann. Der Code des CRTM ist Teil des BIOS und wird nach dem Einschalten des Systems ausgeführt. Hierbei werden Prüfsummen der am Start des Systems beteiligten Komponenten gebildet.

Static Root of Trust for Measurement

Das Static Root of Trust for Measurement (S-RTM) ist das statische RTM. Die Integritätswerte des S-RTM werden in den PCRs 0 bis 15 und 23 abgelegt.

Storage Root Key

Ein asymmetrischer 2048-Bit RSA-Schlüssel innerhalb des TPMs, der für die Verschlüsselung von TPM-internen Daten verwendet wird. Der SRK wird durch Besitzübernahme des TPMs erzeugt und verbleibt permanent innerhalb des TPMs, bis der aktuelle Besitzer wieder gelöscht wird.

TCG Device Driver Library

Software-Bibliothek des TPM Treibers.

TCG Device Driver Library Interface

Die Software-Schnittstelle zum TDDL.

TCG Software Stack

Der Software Stack, der durch die TCG spezifiziert wurde und eine Abstraktionsschicht des TPM ist. Er soll Anwendungen den Zugang zum TPM und dessen Funktionalität ermöglichen und erleichtern.

TPM Device Driver

Der Treiber im Betriebssystem, der benötigt wird um mit dem TPM zu kommunizieren.

TPM Interface Specification

Die Spezifikation der Hardware-Schnittstelle für das TPM Version 1.2.

Trusted Computing

Komponenten und Mechanismen die durch die TCG Spezifikation definiert wurden, um die Vertrauenswürdigkeit von IT-Systemen erhöhen.

Trusted Computing Group

Ein Industriekonsortium, welches verschiedene Spezifikationen definiert, die benötigt werden, um eine vertrauenswürdige Rechnerplattform zu ermöglichen, u.a. die TPM-Spezifikation, die TSS-Spezifikation und die TNC-Spezifikation.

Trusted Network Connect

Die TNC-Architektur konzentriert sich auf Sicherheitslösungen zur Netzwerkzugriffs-kontrolle mit Hilfe von Trusted Computing. Hierzu werden Integritätsmessungen als Sicherheitsbeweis zwischen den Kommunikationsendpunkten ausgetauscht und verifizert, bevor Zugang zum Netzwerk / Zugriff auf Daten im Netzwerk gestattet wird.

Trusted Platform Module

Ein optional nutzbarer, manipulationsgeschützter Hardware-Baustein, der dem Benutzer geschützte Funktionalitäten und isolierten Speicher bereitstellt. Ein TPM verhält sich passiv und beinhaltet z. B. einen Zufallszahlengenerator, einen RSA-Schlüsselgenerator sowie eine Funktion zur Hashwertbildung. Damit kann eine Plattform das TPM aufrufen um Schlüssel zu erzeugen und zu speichern, Daten zu signieren, diese an eine Plattform zu binden oder den Zustand der Plattform zu messen.

TSS Core Service

Der TCG Software Stack gewährt allen TSP s Zugriff auf seine TPM-Funktionen.

TSS Service Provider

Anwendungen, die auf Dienste des TSS zugreifen möchten, nutzen hierfür jeweils einen eigenen TSP.

TSS Service Provider Interface

Die Software-Schnittstelle zum TSP.

Unified Modeling Language

Eine von der Object Management Group (OMG) entwickelte und standardisierte Sprache für die Modellierung von Software. Im Sinne einer Sprache definiert die UML dabei Bezeichner für die meisten Begriffe, die für die Modellierung wichtig sind, und legt mögliche Beziehungen zwischen diesen Begriffen fest. Die UML definiert weiter grafische Notationen für diese Begriffe und für Modelle von statischen Strukturen und von dynamischen Abläufen, die man mit diesen Begriffen formulieren kann.

Virtual Machine

Eine Virtual Machine (VM) ist eine Software-Implementierung eines Computers oder einer Maschine, die sich aus der Sicht eines Betriebssystems wie physikalische Hardware verhält. Virtuelle Maschinen benötigen die Präsenz einer Softwareschicht (VMM), um auf die gemultiplexte physikalische Hardware zuzugreifen.

Virtual Machine Monitor

Ein Virtual Machine Monitor (auch Hypervisor genannt) erzeugt und verwaltet virtuelle Maschinen (VM). Er verteilt die Ressourcen der zugrundeliegenden Hardware derart, dass für den Betrieb jeder einzelnen virtuellen Maschine Ressourcen zur Verfügung stehen.